

AFRL-IF-RS-TR-2005-59 Vol. 1 (of 4)
Interim Report
February 2005



OPEN RADIO COMMUNICATIONS ARCHITECTURE CORE FRAMEWORK V1.1.0 VOLUME 1 SOFTWARE USERS MANUAL

L-3 Communications Government Services, Incorporated

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

Copyright ©2004, L-3 Communications Government Services Inc.

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

STINFO FINAL REPORT

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2005-59 Vol. 1 (of 4) has been reviewed and is approved for publication

APPROVED: /s/

RICHARD D. HINMAN
Project Engineer

FOR THE DIRECTOR: /s/

WARREN H. DEBANY, JR., Technical Advisor
Information Grid Division
Information Directorate

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 074-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503				
1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE FEBRUARY 2005	3. REPORT TYPE AND DATES COVERED Interim Feb 04 – Sep 04		
4. TITLE AND SUBTITLE OPEN RADIO COMMUNICATIONS ARCHITECTURE CORE FRAMEWORK V1.1.0 VOLUME 1 SOFTWARE USERS MANUAL		5. FUNDING NUMBERS C - F30602-01-C-0205 PE - 62702F PR - APAW TA - 02 WU - 01		
6. AUTHOR(S) Mike Gudaitis, Dave Hallatt, A. Bagdasarova, Mike Yax				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) L-3 Communications Government Services, Incorporated 1300-B Floyd Avenue Rome New York 13440		8. PERFORMING ORGANIZATION REPORT NUMBER N/A		
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory/IFG 525 Brooks Road Rome New York 13441-4505		10. SPONSORING / MONITORING AGENCY REPORT NUMBER AFRL-IF-RS-TR-2005-59 Vol. 1 (of 4)		
11. SUPPLEMENTARY NOTES AFRL Project Engineer: Richard D. Hinman/IFG/(315) 330-3616/ Richard.Hinman@rl.af.mil				
12a. DISTRIBUTION / AVAILABILITY STATEMENT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.			12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 Words) This document describes software developed to support the Joint Tactical Radio System (JTRS) program. The software implementation includes a Core Framework (CF) and sample applications that are based on the Software Communications Architecture (SCA) v2.2. The software was designed for a desktop computer running the Linux operating system (OS). It was developed in C++, uses ACE/TAO for CORBA middleware, Xerces for the XML parser, and Red Hat Linux for the Operating System. The software is referred to as, Open Radio Communication Architecture Core Framework, "OrcaCF" (formerly known as LinuxFC), this document describes version 1.1.0 of the OrcaCF. This Software User Manual (SUM) tells a hands-on software user how to install and use the OrcaCF v1.1.0 subsystem. The architecture and requirements are based on the JTRS SCA v2.2.				
14. SUBJECT TERMS Communication, Joint Tactical Radio Systems, JTRS, Software Communication Architecture, SCA, Core Framework, CORBA, Middleware			15. NUMBER OF PAGES 159	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL	

Table of Contents

1.0 SCOPE	1
1.1 Identification	1
1.2 Subsystem Overview	1
1.3 Document Overview.....	3
2.0 REFERENCED DOCUMENTS	4
2.1 Government Documents.....	4
2.2 Non-Government Documents.....	4
3.0 SOFTWARE SUMMARY	5
3.1 Software Application	5
3.2 Software Inventory	5
3.2.1 Contents of the OrcaCF.....	5
3.2.2 Changes Installed.....	5
3.2.3 Related Documents	5
3.2.4 Troubleshooting Tips.....	5
3.2.5 Environment.....	7
3.3 Software Organization and Overview of Operation	8
3.3.1 Naming Service.....	8
3.3.2 Domain Booter.....	8
3.3.3 Node Booter	8
3.3.4 Event Viewer	8
3.3.5 Log Viewer	9
3.3.6 Application HMI.....	9
3.4 Contingencies and Alternate States and Modes of Operation	9
3.5 Requirements Traceability.....	9
3.6 Qualification Provisions	10
3.7 Security and Privacy.....	10
3.8 Feedback	10
3.8.1 Feedback Form.....	11
4.0 ACCESS TO THE SOFTWARE.....	12
4.1 First-time User of the Software.....	12
4.1.1 Equipment Familiarization.....	12
4.1.2 Access Control.....	13
4.1.3 Installation and Setup.....	14
4.2 Initiating a Session	16
4.2.1 Running the Naming Service	17
4.2.2 Running the Domain Booter (CoreFramework/Server).....	18
4.2.3 Running the Node Booter (DeviceManager/Server).....	20
4.2.4 Running the Event Viewer	23
4.2.5 Running the Log Viewer.....	24
4.2.6 Running the Application HMI (Human Machine Interface).....	26
4.3 Stopping and Suspending Work	33
4.4 Uninstalling the Application	34
5.0 SOFTWARE SUPPORT INFORMATION	34
5.1 “As Built” Software Design	34

6.0 COPYRIGHT NOTICE	34
7.0 NOTES	36
Appendix A. License Information	43
Appendix B. Developers Notes.....	70
Appendix C. Compilation Build Procedures	83
Appendix D. Software Design	96
Appendix E. Errata.....	126
Appendix F. Release Notes.....	132
Appendix G. Code Style Guide	134

List of Figures

Figure 1-1 OrcaCF Conceptual View	1
Figure 1-2 SCA Class Structure of the OrcaCF (<i>adapted from Figure 3-3, MSRC-5000SCA v. 2.2</i>).	2
Figure 1-3 Conceptual View of the SoundDemo “Waveform”	3
Figure 3-1 Sound Demo Sate Diagram	9
Figure 4-1 Example of the OrcaCF Directory Structure	16
Figure 4-2 Example Screenshot Showing Start of Naming Service	18
Figure 4-3 Example Screenshot Showing Start of the Domain Booter	19
Figure 4-4 Example Screenshot of NodeBooter Startup	21
Figure 4-5 Example Screenshot Showing DomainBooter after NodeBooter Startup	22
Figure 4-6 Example Screenshot Showing Start of the Event Viewer	23
Figure 4-7 Example Screenshot Showing Start of the Log Viewer	25
Figure 4-8 Example Screenshot Showing Log Viewer after Log selection	25
Figure 4-9 Example Screenshot Showing Application HMI Startup	27
Figure 4-10 Example Screenshot Showing Application HMI After <i>Application</i> Creation	27
Figure 4-11 Example Screenshot Showing NodeBooter After <i>Application</i> Creation	28
Figure 4-12 Example Screenshot Showing DomainBooter After <i>Application</i> Creation	29
Figure 4-13 Example Screenshot Showing Event Viewer After <i>Application</i> Creation	29
Figure 4-14 Example Screenshot Showing Log Activity in the Log Viewer	30
Figure 4-15 KMix toolbar for adjusting audio I/O.	31
Figure 4-16 Example Screenshot Showing Application HMI After User Interaction	32

List of Tables

Table 3-1 File and Directory Structure	5
--	---

1.0 Scope

This document describes software developed to support the Joint Tactical Radio System (JTRS) program. The software implementation includes a Core Framework (CF) and sample applications that are based on the Software Communications Architecture (SCA) v2.2. The software was designed for a desktop computer running the Linux operating system (OS). It was developed in C++, uses ACE/TAO for CORBA middleware, Xerces for the XML parser, and Red Hat Linux for the Operating System. The software is referred to as “OrcaCF” (formerly known as LinuxCF), this document describes version 1.1.0 of the OrcaCF.

1.1 IDENTIFICATION

This Software User Manual (SUM) tells a hands-on software user how to install and use the OrcaCF v1.1.0 subsystem. The architecture and requirements are based on the JTRS SCA v2.2.

1.2 SUBSYSTEM OVERVIEW

This SUM addresses v1.1.0 of the OrcaCF Project. The OrcaCF software was developed to be in compliance with the SCA v2.2 (reference 2.1.a). Specifically, the OrcaCF software contains:

- An Operating System (Linux) per SCA v2.2, Section 3.1.1
- Middleware and Services (ACE/TAO) per SCA v2.2, Section 3.1.2
- A CF per SCA v2.2, Section 3.1.3
- A simple application per SCA v2.2, Section 3.2

and meets Logical Device and General Software Rule requirements of SCA v2.2, Sections 3.3 and 3.4, respectively. Figure 1-1 shows conceptually how the OrcaCF components fit together.

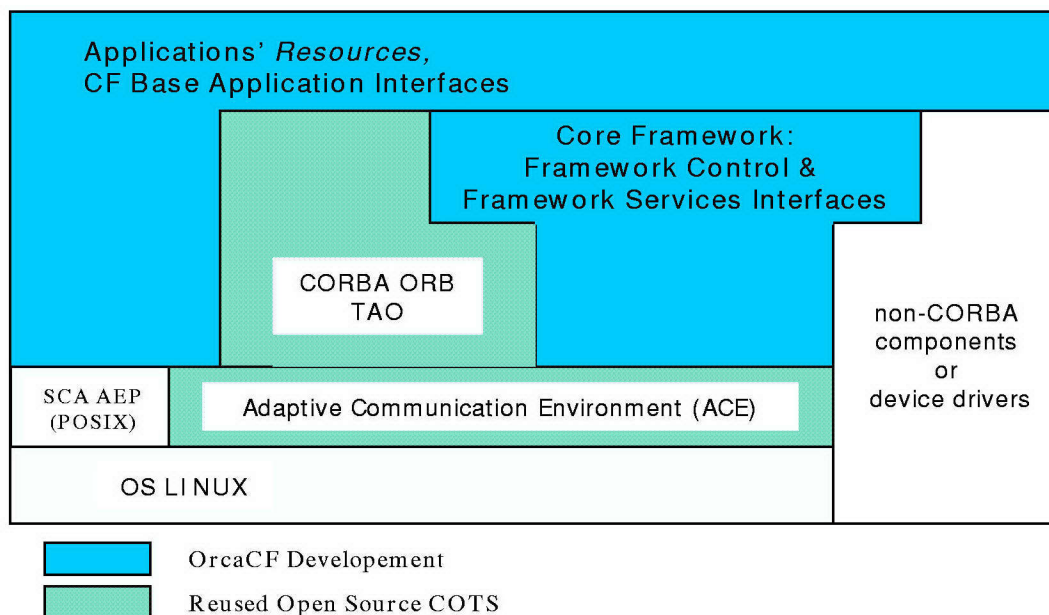


Figure 1-1 OrcaCF Conceptual View

The Linux OS is from Red Hat. The Object Request Broker (ORB) is The ACE ORB (TAO) from Doug Schmidt's web site (reference 2.2.b). The Core Framework and sample applications (blue highlighted areas) is new software development, as described in this SUM. The OrcaCF is developed to run on a

standard Intel x86-based PC. It represents a complete CF implementation that has been tested for compliance to the SCA v2.2 specification. The class structure of the OrcaCF is shown in Figure 1-2 below.

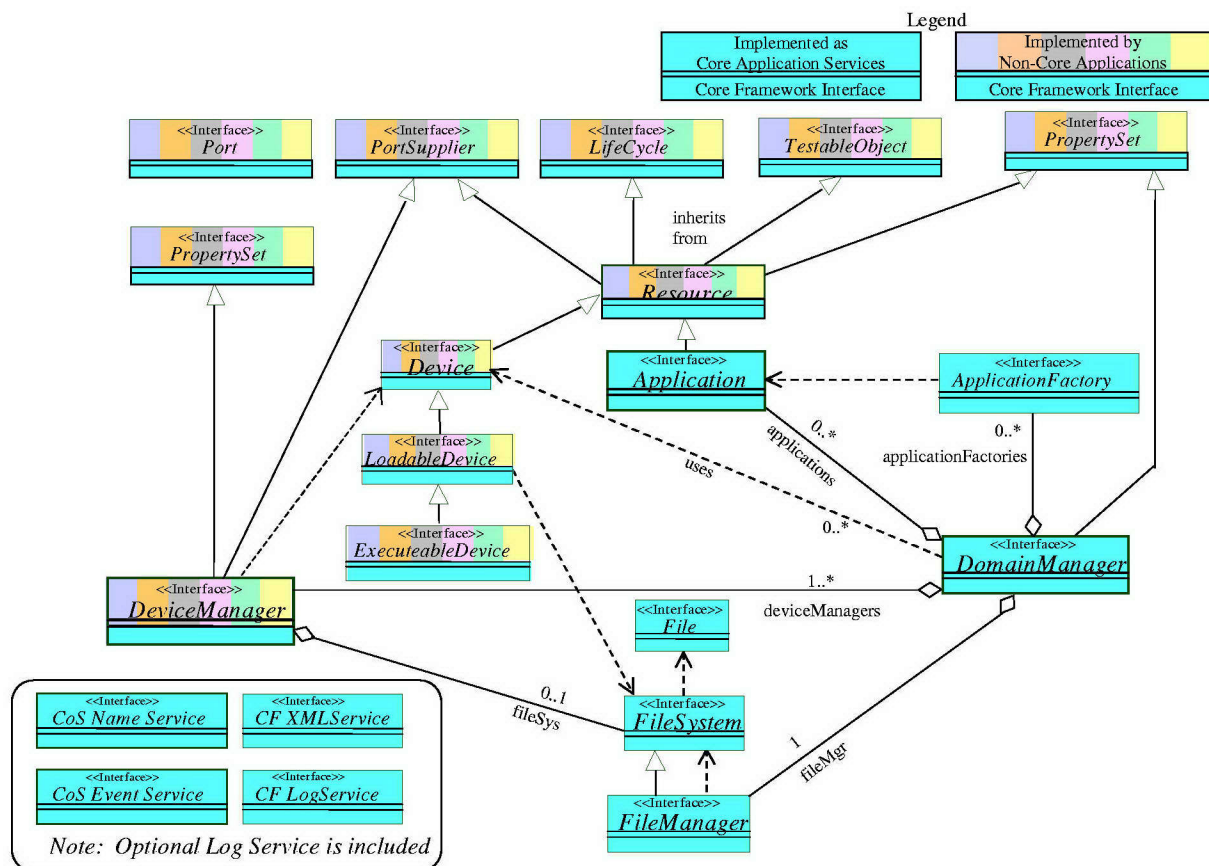


Figure 1-2 SCA Class Structure of the OrcaCF
(adapted from Figure 3-3, MSRC-5000SCA v. 2.2)

The OrcaCF is ideal for rapid prototyping of waveforms built to SCA specifications since it is PC-based, and uses Open Source software components. It has been built and tested on Red Hat 7.3, 9.0, and Fedora Core 1. Red Hat 9.0 was used for the screenshots in the figures in this SUM.

This Release contains one *Application* “waveform”:

1. A simple audio recorder *Application* “waveform” called SoundDemo that will be used to demonstrate the capabilities of the OrcaCF.

The main text of this SUM is written for demonstrating the audio recorder application. The audio recorder *Application*, or SoundDemo “waveform”, has two modes. In the RECORD mode, voice is sampled from the microphone and written to a Sound File; in the PLAYBACK mode, the Sound File is read and the voice recording is played back on the speakers. The SoundDemo “waveform” *Application* consists of an Assembly Controller, a Recorder Resource, and an Application HMI (Human Machine Interface). See Figure 1-3 below for a conceptual view of the SoundDemo “waveform”.

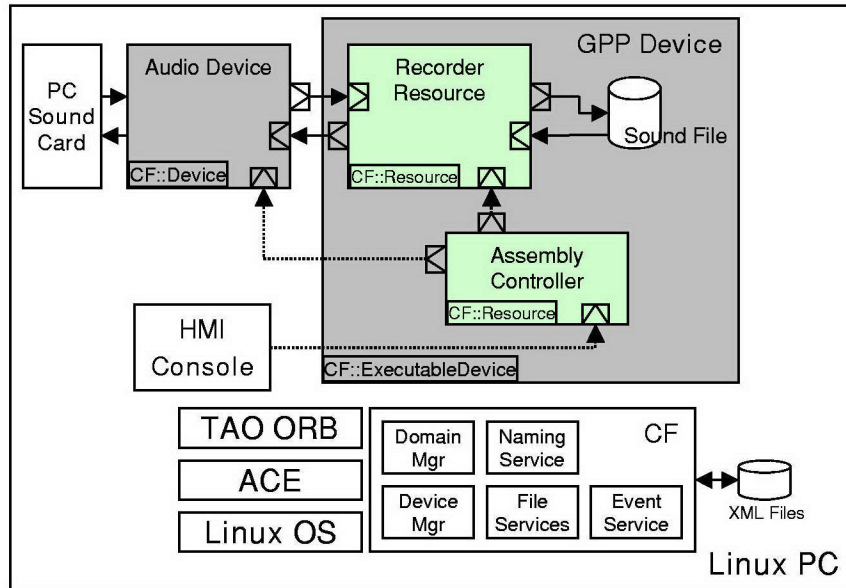


Figure 1-3 Conceptual View of the SoundDemo “Waveform”

1.3 DOCUMENT OVERVIEW

This document is based on a tailored version of Data Item Description (DID) DI-IPSC-81443, SUM, combined with applicable sections of DI-IPSC-81442, Software Version Description (SVD), DI-IPSC-81439, Software Test Description (STD), and DI-IPSC-81441, Software Product Specification (SPS). For the OrcaCF v1.1.0, these separate DIDs would contain redundant information. Therefore, they were combined into one document to simplify configuration management and provide consistent information to the user. This SUM shall be maintained and modified to reflect the current build of the OrcaCF Project.

Section 2.0 lists the documents referenced by this SUM and used during its preparation.

Section 3.0 provides a summary of the delivered software.

Section 4.0 includes step-by-step procedures oriented to the first time/occasional user to test the functionality of the delivered software.

Section 5.0 references Software Design.

Section 6.0 addresses the distribution, licensing and copyright issues.

Section 7.0 provides notes to aid in the general understanding of this document and the software application.

2.0 REFERENCED DOCUMENTS

2.1 GOVERNMENT DOCUMENTS

Standards and other publications produced by government agencies that have been utilized in creating this SUM and may also be utilized while developing the OrcaCF software product are listed here.

- a. Software Communication Architecture (SCA) Specification with Appendices, MSRC-5000SCA V2.2, 17 November 2001
- b. Application Program Interface (API) Supplement to the Software Communications Architecture Specification with Appendices, MSRC-5000API V3.0, 17 November 2001
- c. Security Supplement to the Software Communications Architecture Specification with Appendices, MSRC-5000API V3.0, 17 November 2001
- d. Support and Rationale Document (SRD) for the Software Communication Architecture Specification, V2.2, 19 December 2001.
- e. JTeL SCA Requirements Matrix, Export+V1+-+Load+v1+-+SCA+Baseline+ReqmtsV3.0.xls, 8 Aug 2002
- f. Process Asset Library (PAL), Systems Engineering Process Office (SEPO), SPAWAR Systems Center, <http://sepo.spawar.navy.mil/sepo/index2.html>
- g. JTeL Virtual Private Office, <https://vpo.spawar.navy.mil/SSC-SD/JTRS/TL/master.nsf> (password protected)
- h. OrcaCF Software Requirements Specification (SRS), OrcaCF_SRS_v1_1_0.doc, Jun 2004.

2.2 NON-GOVERNMENT DOCUMENTS

Same as previous subsection but the documents were not published by government agencies.

- a. Red Hat Linux website: <http://www.redhat.com>
- b. ACE/TAO websites: <http://www.cs.wustl.edu/~schmidt/TAO.html>, <http://www.theaceorb.com/>
- c. OCI TAO Developers Guide version 1.2a, volume 1&2 (Part numbers 510-01, 510-02), Object Computing Inc., 2002.
- d. Software Engineering Institute Capability Maturity Model for Software, Version 3.0, February 1993
- e. Industry Implementation of International Standard ISO/IEC 12207: 1995, Standard for Information Technology, Software Life Cycle Processes, IEEE/EIA 12207 Series, March 1998
- f. OMG Software Based Communication Domain Task Force, <http://sbc.omg.org>
- g. Pure CORBA by Fintan Bolton, 5th Edition, ISBN 0-672-321812. Sam's Publishing Inc., 2002.
- h. Advanced CORBA Programming with C++, by Michi Henning & Steve Vinoski, ISBN 0-201-37927-9, Addison-Wesley Longman Inc., 1999.
- i. Open Sound System API documentation: <http://www.opensound.com/pguide/index.html>.

3.0 Software Summary

3.1 SOFTWARE APPLICATION

The OrcaCF v1.1.0 is a Core Framework developed and tested in accordance with SCA v2.2. It includes sample applications for running and testing the current CF development.

3.2 SOFTWARE INVENTORY

All documents, files and programs for running the OrcaCF v1.1.0 application are included in the zipped file provided with this document. The SUM refers to the *binary* distribution of the OrcaCF v1.1.0. For information and instructions for the *source* distribution of the OrcaCF v1.1.0, please refer to the OrcaCF_SUM_App_C_CompilationBuildProcedures_v1_1_0.doc. A description of all materials included in the binary distribution is provided in section 3.2.1.

3.2.1 CONTENTS OF THE ORCACF

Table 3-1 provides a description of the file structure and contents of the zipped file.

Directory	Description
OrcaCF	All of the application scripts and a Quick Reference Guide (QRG) for running the software.
OrcaCF/bin	Binaries required for running the application.
OrcaCF/doc	Documentation
OrcaCF/inc	Required header source files and generated source code.
OrcaCF/lib	Required shared library files.
OrcaCF/projects	All developed and generated source code.
OrcaCF/tmp	A temporary directory used during CoreFramework operation.
OrcaCF/xml	XML files required for configuration of the OrcaCF components.

Table 3-1 File and Directory Structure

3.2.2 CHANGES INSTALLED

This is the first formal release of the OrcaCF. See Appendix F of this SUM, entitled Release Notes, for a list of SCA change proposals (CPs) incorporated in this release.

3.2.3 RELATED DOCUMENTS

Refer to Section 1.3 concerning related documents.

3.2.4 TROUBLESHOOTING TIPS

This section provides a listing of known issues pertaining to the running of the OrcaCF. For issues regarding the SCA v2.2, change proposals and requirements, refer to the Errata included in Appendix G.

1. This Release has only been verified on hardware configurations that include SoundBlaster Live!, and SoundBlaster PCI128 cards, as well as motherboard sound devices consisting of the Analog Devices (AD1885) AC 97 Codec. However with motherboard sound, you may encounter write errors or read errors if you press the record and play keystrokes in rapid succession. The sound device supported by your PC will be displayed in KMiX above the slider bars. We have not verified the OrcaCF Sound Demo operation with other soundcards or on-board sound devices.

2. Attempts to execute the OrcaCF scripts may result in a “*permission denied*” error. If such an error occurs, verify that you have “execute” permissions for the scripts. This may be done by locating the scripts in your file browser of choice, right-clicking the script and selecting Properties, and verifying the appropriate check box is selected within the Permissions tab. This same “*permission denied*” error may also result due to improper permissions assigned to the actual executables, which are located in the `/home/<username>/OrcaCF/bin` directory. Locate the executables and verify that you have “execute” permissions for those files.
3. The OrcaCF directory resulting from unzipping the `OrcaCF_v1_1_0_binary` package includes a `tmp` subdirectory. Even though it is empty upon unzipping the package, it is not to be deleted. Deleting this `tmp` subdirectory will cause a file exception error to occur during execution of the “`startDomainBooter`” executable script. This `tmp` subdirectory is used by the CoreFramework.
4. Running the OrcaCF requires the use of a `GPPTemp` directory with “write” and “execute” permissions. The final path of this directory is `/home/<username>/OrcaCF/tmp/GPPTemp`. If this `GPPTemp` directory does not exist prior to testing OrcaCF, it will be created during OrcaCF execution with you as the owner, and you will be given all necessary permissions for this directory. If this folder does exist prior to testing OrcaCF, you must verify that you have “write” and “execute” permissions. A consequence of not having “write” permissions is a “Create Application Error” during execution of the “`startApplicationHMI`” executable script. A consequence of not having “execute” permissions is a “Caught a CORBA exception” error during execution of the “`startApplicationHMI`” executable script.
5. Running the Sound Demo requires the user to verify that the microphone is set as the recording device. Red Hat Linux 9.0 contains a mixer program, `KMix`, which lets you modify volume settings for the sound device, and select the desired recording device. The currently active recording device is indicated by a red light beneath the device’s volume control. Make sure the red light is lit below the microphone volume control prior to running the `SoundDemo`. If the sound test is performed without the microphone as the recording device, the result will be the absence of sound, or noisy sound without voice during playback.
6. The `AudioDevice` interface was programmed using the Open Sound System (OSS) API (reference 2.2.i), which is the standard sound API for Red Hat Linux 9.0 and earlier. The default configuration values for the sound card are shown in the `NodeBooter` window. If you have problems with the sound performance, check the settings displayed in the `NodeBooter` window. For example, using integrated motherboard sound, the following text may be seen:

```
[AUDIODEV]: Configure: BITS/SAMPLE - 16
[AUDIODEV]: Configure: MONO/STEREO - Stereo
[AUDIODEV]: Configure: SAMPLE RATE(Hz) - 16000
[AUDIODEV]: Configure: BLOCK SIZE - 16384
[AUDIODEV]: Configure: BUFFER SIZE - 32768
```

The first three values are set in the XML files. These are the default values used by the OrcaCF to configure the `AudioDevice`. The `BLOCK SIZE` is the audio buffer memory used by the sound card. It is specified in bytes and allocated in RAM for getting data to/from the soundcard. The `BUFFER SIZE` refers to the size of the CORBA packet used by the OrcaCF to move the audio data between CORBA objects. The `BUFFER SIZE` must be at least twice as large as the `BLOCK SIZE` for proper performance.

7. This distribution has been tested on Red Hat Linux 7.3, 9.0, and Fedora Core 1. The `Sound Demo` does not run properly under Fedora Core 2, which is the latest free distribution sponsored

by Red Hat. Fedora Core 2 uses the Linux kernel 2.6.x which removed native support for the Open Sound System (OSS), and switched the default to Advanced Linux Sound Architecture (ALSA). The OrcaCF Sound Demo uses OSS which doesn't work well with the current version of ALSA.

8. Errors and exceptions that may occur while testing OrcaCF v1.1.0 are often caused because a duplicate of the process attempting execution is already running in the background. If an error occurs while running the core framework tests, troubleshooting should begin by checking all currently running processes using the "top" utility. In a console window enter `top` at the prompt and press `<ENTER>`. Type `u`, then your `<username>`, and then press `<ENTER>` to identify any OrcaCF processes running. Terminate all OrcaCF processes before retesting the core framework. Terminating a process is done within the `top` utility by typing `k`, hitting `<ENTER>`, and entering the process ID number, and then hitting `<ENTER>` *twice*.
9. Check the website, www.OrcaCF.com for the latest information.

3.2.5 ENVIRONMENT

OrcaCF v1.1.0 is delivered with all the necessary binary executable software the user needs to run the software. The software environment is Red Hat Linux 9.0 running on a standard x86 PC. Other versions of Linux can be supported by recompiling the source code on the target OS.

3.2.5.1 HARDWARE

The following is the minimum system configuration recommended for reliable performance of the OrcaCF v1.1.0.

- CPU – Pentium III 550MHz
- RAM – 512MB SDRAM
- Display – ATI RAGE 128
- Sound – SoundBlaster PCI 128
- Storage – 10GB IDE hard drive with ext3 filesystem
- NIC - 3Com 3C590/3C595/3C90x

3.2.5.2 SOFTWARE

The following software packages, along with their version numbers, were installed on the Linux testing machine and is the minimum configuration needed for reliable performance. The binaries delivered with this distribution will only work for Red Hat Linux 9.0.

- Desktop – Red Hat Linux 9.0
 - Linux *Kernel 2.4.20-8
 - KDE 3.1-10
- Properly configured sound card – (Our configuration listed below)
 - Sound Driver – ES1371 AudioPCI97 Driver v0.31

**Note: Linux kernel versions 2.6.x. changed audio support from OSS to ALSA. The Sound Demo does not work with ALSA in the 2.6.x kernel used in Fedora Core 2.*

Listed below is the software used in *developing* the OrcaCF, but not required to run the demo:

- IDE – KDevelop 2.1.5
- Compiler – gcc 3.2.2
- make 3.79.1
- ACE v5.4 / TAO v1.4
- XML Parser – Apache Xerces v2.5.0
- Net/File Browser – Konqueror 3.0.0-12

3.3 SOFTWARE ORGANIZATION AND OVERVIEW OF OPERATION

OrcaCF v1.1.0 consists of six executables running in separate processes. Each process has its own ORB instantiation. The five executable processes are referred to as Naming Service, Domain Booter, Node Booter, Event Viewer, Log Viewer and Application Human Machine Interface (HMI). Each of these will be explained in more detail below.

3.3.1 NAMING SERVICE

The Naming Service executable included with the OrcaCF v1.1.0 is ACE/TAO's open source implementation of the Object Management Group's (OMG) Common Object Request Broker Architecture (CORBA) Naming Service Specification. The Naming Service allows CORBA objects to be associated with an abstract name, which can be used by CORBA clients to locate the objects. A script file is provided to start the Naming Service executable. It is essential that this Naming Service is started first and is running before any of the other scripts are executed.

3.3.2 DOMAIN BOOTER

The Domain Booter is an executable that starts and runs the Domain Manager and Event Service. A script file is provided to start the Domain Booter executable. Upon startup, the Domain Booter locates the Naming Service and creates the DomainManager. The DomainManager then parses its DomainManager Configuration Descriptor (DMD) Extensible Markup Language (XML) file to determine its Naming Service name. It then registers itself with the Naming Service using the Naming Service name it obtained from the DMD. This registration allows CORBA clients to find the DomainManager via the Naming Service. It is essential that the Domain Booter is started *after* the Naming Service, and is running before any of the remaining executables are started. If the Domain Booter is stopped, the Naming Service must be restarted prior to restarting the Domain Booter.

3.3.3 NODE BOOTER

The Node Booter is an executable that starts up the DeviceManager and any of its initial devices (AudioDevice, GPPDevice) and services (Log Service) that are listed in the Device Configuration Descriptor (DCD) XML file. A script file is provided to start the Node Booter executable. The components listed in the DCD can be Devices and/or Services. Any initial connections between components listed in the DCD are made by the DomainManager after the DeviceManager registers with the DomainManager.

3.3.4 EVENT VIEWER

The Event Viewer is a utility program that allows the user to monitor events being passed to the Incoming Domain Management (IDM) Event Channel (IDM_Channel) and Outgoing Domain Management (ODM) Event Channel (ODM_Channel) (See SCA v.2.2, Section 3.1.2.4.1). A script file is provided to start the Event Viewer executable. The Event Viewer is optional and is not required to run the rest of the executables. In order to view events generated by any component, the Event Viewer must be running

prior to the component's execution. For the Event Viewer to function properly, the Naming Service and Domain Booter must be running.

3.3.5 LOG VIEWER

The Log Viewer is a utility program that allows the user to monitor Logs that are being written by components. A script file is provided to start the Log Viewer executable. The Log Viewer is optional and is not required to run the rest of the executables. In order to view Logs generated by any component, the Log Viewer must be running prior to the component's executable. For the Log Viewer to function properly, the Naming Service, Domain Booter, and Node Booter must be running.

3.3.6 APPLICATION HMI

The Application HMI is a User Interface (UI) utility program that allows a user to select the Sound Demo. The remainder of this document refers to instructions for running the Sound Demo.

3.3.6.1 STATE CHART OF SOUND DEMO

Figure 3-1 below depicts the possible states for the Sound Demo HMI. These states include Record, Play, Stop and Quit. The states of the Sound Demo are changed based on the keyboard input from the user.

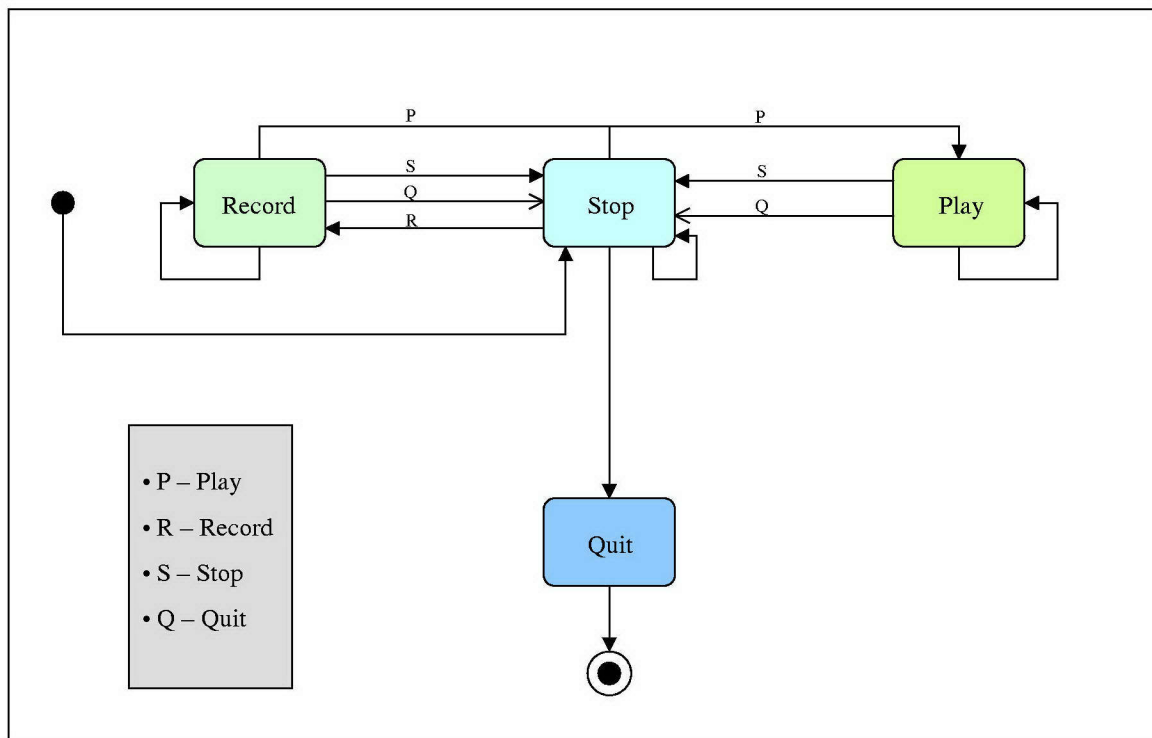


Figure 3-1 Sound Demo State Diagram

3.4 CONTINGENCIES AND ALTERNATE STATES AND MODES OF OPERATION

N/A

3.5 REQUIREMENTS TRACEABILITY

For requirements traceability, the Software Requirements Specification (SRS) is provided. Refer to that document for additional information regarding the implemented requirements for v1.1.0.

3.6 QUALIFICATION PROVISIONS

The JTAP test tool was utilized to perform testing and compliance of the OrcaCF with the requirements identified in Section 3.5 above. Test Results are available by official request. Send your request to OrcaCF.gsi@L-3com.com.

3.7 SECURITY AND PRIVACY

There are currently no security or privacy issues associated with v1.1.0 of the OrcaCF. See Appendix A to this SUM, entitled License Information, for complete licensing and copyright information associated with the v1.1.0 release of the OrcaCF.

3.8 FEEDBACK

We welcome and encourage any feedback or comments regarding this software. Although the OrcaCF is provided as is, without any implied or explicit support, your feedback may be used to improve the software in a future release. For your convenience we have provided a form on the following page. You may also contact any of the Software Engineers listed below:

Mike Gudaitis

Email: Mike.Gudaitis@l-3com.com.

-OR-

David Hallatt

Email: Dave.Hallatt@l-3com.com.

-OR-

Doug Ackerman

Email: Douglas.Ackerman@l-3com.com.

Mailing Address:

L-3 Communications Government Services Inc.

1300B Floyd Avenue, Rome NY 13440

Phone: 315-339-6184

Fax: 315-339-6923

Instructions:

Your Contact Information

Computer Environment

Feedback pertains to:

Comments:

--

4.0 Access to the software

This section contains step-by-step procedures oriented to the first-time/occasional user. Enough detail is presented for the user to reliably access the software before learning the details of its functional capabilities.

4.1 FIRST-TIME USER OF THE SOFTWARE

The following paragraphs provide the instructions to run the v1.1.0 software application.

4.1.1 EQUIPMENT FAMILIARIZATION

This software requires a standard x86 PC. It is assumed that the user has a working knowledge of how to operate a standard PC. The user should refer to the appropriate computer manual for the following information:

- a. Procedures for turning on power and making adjustments.
- b. Dimensions and capabilities of the visual display screen.
- c. Appearance of the cursor, how to identify an active cursor if more than one cursor can appear, how to position a cursor, and how to use a cursor.
- d. Keyboard layout and role of different types of keys and pointing devices.
- e. Procedures for turning power off if special sequencing of operations is needed.

4.1.1.1 INFORMATION ON LINUX ENVIRONMENT

The section below is included for those who may not be familiar with certain software, or software concepts, used for the OrcaCF. This section includes references for further research of a particular topic.

This software has been developed on a PC utilizing the KDE desktop that comes with Red Hat Linux. The default desktop for most Red Hat Linux installations is the GNOME desktop. The underlying functionality of Linux applications works the same for each desktop, so you should have no problem using either one. Website links are provided below if you have any questions regarding use of either desktop.

- KDE website: <http://www.kde.org/>
- GNOME website: <http://www.gnome.org/>

The bash shell is the default shell when installing most Red Hat Linux distributions. A shell is basically a command language interpreter. When you open a terminal window (or shell) in Linux, it defaults to the bash shell. As the terminal window comes up, it reads certain bash specific configuration files. This terminal window will now only recognize and interpret commands according to the bash shell. It is important to note that the OrcaCF test scripts were done using the bash shell. You can find more information about the bash shell at its website, listed below.

- Bash Reference Manual: <http://www.gnu.org/manual/bash/index.html>
- University of Washington – Shells & Shell Scripts:
<http://www.washington.edu/computing/unix/shell.html>

Some user environment variables need to be configured to run the OrcaCF scenarios. It is assumed the user has a working knowledge of the Linux/Unix operating system, as well as standard programming principles, such as setting environment variables. Red Hat Linux 7.3 has an easy to follow wizard that guides the user through a simple setup of their account when they first logon. Below are some helpful web links for beginners to become familiar with these concepts.

- The Single Unix Specification – Environment Variables:
<http://www.opengroup.org/onlinepubs/007908799/xbd/envvar.html>

- Red Hat Linux: <http://www.redhat.com/>

There are a number of different Editors, File managers, and other applications that come with the Red Hat distribution of Linux. The instructions in this document do not advocate using a particular application to perform a particular task. It is up to the user to select the application they prefer. For example, we use the file manager called Konqueror when browsing for files. Another user may prefer to open a terminal window and use the command line to search for files. The instructions described below are the methods we chose to use, but you may freely choose other methods for performing the same tasks.

The installation steps refer to directories on the computer you are working on and it's important to note that your directories will look different from ours in some instances. We will denote directory differences by using < > characters. For example, when I logon to our Linux machine my home directory looks like: /home/dackerman. I will list your directory as: /home/<username>. You would obviously replace <username> with your actual logon name. Any other directory differences will be represented in a similar fashion.

Note: Linux is case sensitive. Pay attention to caps and extra spaces. Type paths and commands exactly as they appear.

4.1.1.2 INFORMATION ON CORBA

The user should have some knowledge of CORBA in order to understand how this application works. CORBA is the specified standard middleware of the JTRS SCA. A CORBA application has a client and servant. These terms will be used in the procedures described in this section. For general information on CORBA, refer to the OMG website <http://www.omg.org/gettingstarted/corbafaq.htm>. For tutorial information regarding TAO, the specific ORB used in this application, refer to <http://www.cs.wustl.edu/~schmidt/tutorials-corba.html>.

4.1.1.3 INFORMATION ON XML

The user should have some knowledge of XML and XML parsing in order to understand how this application works. XML is used extensively within the SCA for configuration and control of SCA components. For general information on XML, refer to the World Wide Web Consortium (W3C) website <http://www.w3.org/XML/>. For specific information on the XML parser used for the OrcaCF, XERCES, refer to the Apache Software Foundation (Apache) website <http://xml.apache.org/xerces-c/index.html>.

4.1.2 ACCESS CONTROL

Check with your system administrator to set up an account on your computer, and to review the access and security features of your user account. Follow your local procedures to obtain the following information, if applicable:

- a. How and from whom to obtain a password.
- b. How to add, delete, or change passwords under user control.
- c. Security and privacy considerations pertaining to the storage and marking of output reports and other media that the user will generate.

The information generated by this software is unclassified. However, you should follow your local procedures for handling and storage of the information.

4.1.3 INSTALLATION AND SETUP

Below are the steps for installing/setting up OrcaCF:

1. Set up the environment variables required to run this Release on your computer. Open your `.bashrc` file with your preferred editor. Your `.bashrc` file is located in your home directory (`/home/<username>`). If you don't see the `.bashrc` file in your home directory, you may need to select the *Show Hidden Files* option in the file manager you are using. To do this in Konqueror, select View->Show Hidden Files.

Add the following lines at the end of the `.bashrc` file:

- `export ORCACF_ROOT=$HOME/OrcaCF`
- `export PATH=$PATH:$ORCACF_ROOT/bin`
- `export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$ORCACF_ROOT/lib`
- `export NS_OPTIONS="-ORBdottedDecimalAddresses 1 -ORBEndpoint \`
`iiop://<hostname>:<port> -m 0 -d"`
- `export CF_OPTIONS="-ORBdottedDecimalAddresses 1 -ORBInitRef \`
`NameService=corbaloc::<hostname>:<port>/NameService"`

The `NS_OPTIONS` and `CF_OPTIONS` environment variables contain parameters that are unique and must be set by the user. The `<hostname>` is the `HOSTNAME` of the machine that OrcaCF is executed on and the `<port>` is a unique port number that will be used by OrcaCF executables. The `NS_OPTIONS` contains the `ORBEndpoint` parameter that tells the ORB to listen for requests on the interface specified by the *endpoint*. Endpoints are specified using a URL style format. An example of an IIOP endpoint is:

```
iiop://localhost:9999
```

The standard installation of Linux distributions installs a network LOOPBACK interface called "localhost" with an IP address of 127.0.0.1. Using "localhost" is recommended for anyone not familiar with networking. If you have altered the "localhost" interface in any way, the application may not work properly. The `CF_OPTIONS` environment variable contains the `ORBInitRef` parameter, which is the ORB initial reference argument. This argument allows specification of an arbitrary object reference for an initial service which, in this case, is the Naming Service. The format is:

```
-ORBInitRef [ObjectID]=[ObjectURL]
```

Using "localhost" (recommended), the line would look like this:

```
-ORBInitRef NameService=corbaloc::localhost:9999/NameService
```

The "localhost" and "port" must match for proper CORBA communication.

Here is what a typical `.bashrc` file should look like:

```
# .bashrc

# User specific aliases and functions

# Source global definitions
if [ -f /etc/bashrc ]; then
    . /etc/bashrc
fi

# OrcaCF v1.1.0 ENVIRONMENT VARIABLES
export ORCACF_ROOT=$HOME/OrcaCF
export PATH=$PATH:$ORCACF_ROOT/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$ORCACF_ROOT/lib
export NS_OPTIONS="--ORBottedDecimalAddresses 1 -ORBEndpoint iiop://localhost:9999 -m 0 -d"
export CF_OPTIONS="--ORBottedDecimalAddresses 1 -ORBInitRef \
NameService=corbaloc::localhost:9999/NameService"
```

2. Save the file.
3. After making changes to your `.bashrc` file, you should log out and then log back in, to ensure the changes take effect.
4. Place the `OrcaCF_v1_1_0_binary.tar.gz` file into your home directory.
(`/home/<username>/`)
5. Extract the `OrcaCF_v1_1_0_binary.tar.gz` file. Open a terminal window, go to your `/home/<username>` directory, and type the following:
 - `gunzip OrcaCF_v1_1_0_binary.tar.gz` `<ENTER>`
 - `tar -xvf OrcaCF_v1_1_0_binary.tar` `<ENTER>`
6. You should now have a directory called: `/home/<username>/OrcaCF`
The complete directory tree for the OrcaCF directory should look like the following:

Name	Size	File Type	Modified	Permissions
OrcaCF	4.0 KB	Directory	2004-06-23 10:11	rwxr-xr-x
bin	4.0 KB	Directory	2004-06-14 15:19	rwxr-xr-x
applicationhmi	61.5 KB	Executable File	2004-06-14 15:02	rwxr-xr-x
applications	123 B	Plain Text	2004-06-14 15:01	rw-r--r--
domainbooter	760.4 KB	Executable File	2004-06-14 15:01	rwxr-xr-x
eventviewer	61.9 KB	Executable File	2004-06-14 15:02	rwxr-xr-x
logservice	70.3 KB	Executable File	2004-06-14 15:02	rwxr-xr-x
logviewer	60.4 KB	Executable File	2004-06-14 15:02	rwxr-xr-x
Naming_Service	303.6 KB	Executable File	2004-06-14 15:19	rwxr-xr-x
nodebooter	259.2 KB	Executable File	2004-06-14 15:02	rwxr-xr-x
sounddemoassemblycontroller	123.6 KB	Executable File	2004-06-14 15:03	rwxr-xr-x
sounddemoresource	163.8 KB	Executable File	2004-06-14 15:03	rwxr-xr-x
startApplicationHMI	725 B	Shell Script	2004-06-14 15:02	rwxr-xr-x
startDomainBooter	1.3 KB	Shell Script	2004-06-14 15:01	rwxr-xr-x
startEventViewer	809 B	Shell Script	2004-06-14 15:02	rwxr-xr-x
startLogViewer	801 B	Shell Script	2004-06-14 15:02	rwxr-xr-x
startNamingService	858 B	Shell Script	2004-06-14 15:01	rwxr-xr-x
startNodeBooter	848 B	Shell Script	2004-06-14 15:02	rwxr-xr-x
doc	4.0 KB	Directory	2004-06-15 08:05	rwxr-xr-x
inc	4.0 KB	Directory	2004-06-14 15:02	rwxr-xr-x
lib	4.0 KB	Directory	2004-06-14 15:41	rwxr-xr-x
xml	4.0 KB	Directory	2004-06-14 15:03	rwxr-xr-x
ReadmeFirst_OrcaCF_v1_1_0.txt	11.0 KB	Plain Text	2004-06-23 09:56	rwxr--r--

Figure 4-1 Example of the OrcaCF Directory Structure

7. You are now ready to run the application. See the instructions in the next section.

4.2 INITIATING A SESSION

The OrcaCF package consists of six executables that are run by the user: the CORBA Naming Service, the Domain Booter (CoreFramework/Server), the Node Booter (DeviceManager/Server), the Application HMI, and the optional Event Viewer and Log Viewer. The executables are run using script files. The scripts for running the executables are listed below:

1. startNamingService
2. startDomainBooter
3. StartNodeBooter
4. startEventViewer (*optional*)
5. startLogViewer (*optional*)
6. startApplicationHMI

The order in which these scripts are run is important for proper operation of the OrcaCF. Start each script in sequence, from 1 to 6. If you choose not to run the optional Event Viewer and Log Viewer, you may omit steps 4 and 5. Instructions for running these scripts are described in the sections that follow. Script 1 starts the Naming Service. Script 2 starts (boots up) the Core Framework. Script 3 starts (boots up) the DeviceManager. Scripts 1 through 3 launch the appropriate components and make the necessary connections required to run the Application. Scripts 4 & 5 allow the user to monitor Events and Logs. Script 6 starts the Application. A terminal window must be opened for each script/executable that you plan to run.

****Note:** *BEFORE* you run the Application you should disable the [soundserver](#) on your machine. The [soundserver](#) runs by default when any user logs into KDE. To disable this, open the KDE Control Center and select : Sound->Sound Server. On the General tab uncheck the top box labeled start aRts soundserver on KDE startup. Click Apply and Accept the change. The settings described here might be located in a different place depending on which Linux distribution you have.

When following these instructions for running the v1.1.0 application, it has been assumed that the installation procedures from Section 1.1.1 have been followed.

4.2.1 RUNNING THE NAMING SERVICE

4.2.1.1 PRE-CONDITIONS

Before starting the Naming Service the following pre-condition(s) must be met.

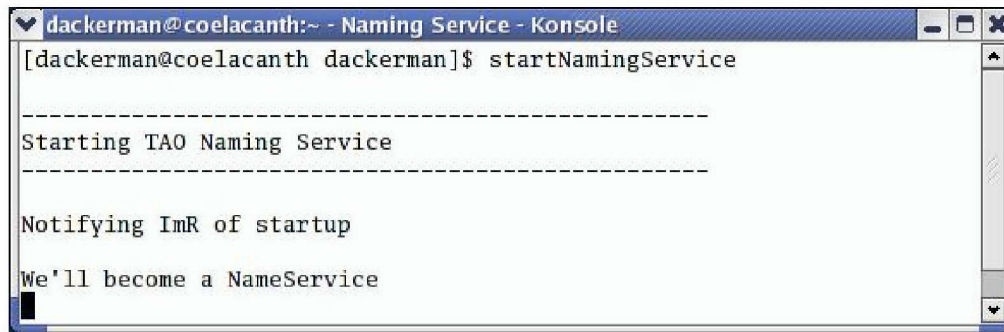
1. In order to run the Naming Service you must ensure that your environment has been setup properly, as previously described in Section 1.1.1.

4.2.1.2 STARTUP

The Naming Service runs in a terminal window. Open up a new terminal window and from the prompt enter the following command: `startNamingService` and hit `<ENTER>`. This will start the Naming Service executable. This executable may need time to complete its own internal boot up process. Do not move on to the next step until the console window output indicates the executable is running. You should see the output shown in Figure 4-2 to ensure the executable is running. If you receive a message indicating *'Permission Denied'* you may have to change the file permissions for the user to have execute privileges. If you experience other problems, check to make sure that the Naming Service is not already started. Type `"top"` and hit `<ENTER>` at the command line to view all processes running. If the Naming Service is listed as a running process, kill the process and start again. Refer to section 3.2.4 Troubleshooting Tips for additional information on changing file permissions or using the `"top"` utility.

4.2.1.3 RUNNING

The following output should be seen within the Terminal Window:



```
dackerman@coelacanth:~ - Naming Service - Konsole
[dackerman@coelacanth dackerman]$ startNamingService

-----
Starting TAO Naming Service
-----

Notifying ImR of startup

We'll become a NameService
```

Figure 4-2 Example Screenshot Showing Start of Naming Service

4.2.2 RUNNING THE DOMAIN BOOTER (COREFRAMEWORK/SERVER)

4.2.2.1 PRE-CONDITIONS

Before starting the DomainBooter the following pre-condition(s) must be met.

1. In order to run the DomainBooter you must ensure that your environment has been setup properly, as previously described in Section 1.1.1.
2. In order to run the DomainBooter the Naming Service must be running, as previously described in Section 4.2.1.

4.2.2.2 STARTUP

The DomainBooter runs in a terminal window. Open up a new terminal window and from the prompt enter the following command: `startDomainBooter` and hit `<ENTER>`. This will start the DomainBooter executable. This executable may need time to complete its own internal boot up process. Do not move on to the next step until the console window output indicates the executable is running. You should see the output shown in Figure 4-3 to ensure the executable is running. If you receive a message indicating *'Permission Denied'* you may have to change the file permissions for the user to have execute privileges. If you experience other problems, check to make sure that the DomainBooter is not already started. Type `"top"` and hit `<ENTER>` at the command line to view all processes running. If the DomainBooter is listed as a running process, kill the process and start again. Refer to section 3.2.4 Troubleshooting Tips for additional information on changing file permissions or using the `"top"` utility.

4.2.2.3 RUNNING

The output shown in Figure 4-3 should be seen within the console for the Domain Booter. The OrcaCF v1.1.0 (CoreFramework/Server) is now running and accepting messages sent to it by client applications. An explanation of the output shown in Figure 4-3 will be provided in the next section 4.2.2.4 Explanation. Stopping the OrcaCF v1.1.0 Server application will be explained in section 4.3 Stopping and Suspending Work.

```
dackerman@coelacanth:~ - Domain Booter - Konsole
[dackerman@coelacanth dackerman]$ startDomainBooter

-----
Starting OrcaCF Domain Booter
-----

[DOMAIN_BOOT]: TAO ORB has been initialized.
[DOMAIN_BOOT]: RootPOA has been initialized.
[DOMAIN_BOOT]: POA Manager has been initialized and activated.
[DOMAIN_BOOT]: Root Naming Context has been obtained.
[DOMAIN_MGR]: SERVICE Not Found - CREATE a Pending Connection.
[DOMAIN_MGR]: LogPort created.
[DOMAIN_MGR]: DomainManager Configured
[DOMAIN_BOOT]: DomainManager has been created

DOMAIN BOOTER COMPLETE.
```

Figure 4-3 Example Screenshot Showing Start of the Domain Booter

4.2.2.4 EXPLANATION

This section provides an explanation of the steps shown in the Figure 4-3. Pre-conditions and a detailed description shall be provided. The pre-conditions are the conditions necessary before execution of the application. The description shall be a detailed explanation of the work being performed during the execution of the application.

Pre-Conditions:

- The Naming Service must be running.

Description:

- The CORBA environment is initialized first. This includes the initialization of the Object Request Broker (ORB), the Portable Object Adapter (POA), and the POA Manager.
- A reference to the Naming Service is obtained.
- A DomainManager C++ object is created.
- The DomainManager creates a CF::FileManager.
- The DomainManager creates a CF::FileSystem.
- The DomainManager mounts its FileSystem to its FileManager with a mountpoint of ORCACF_ROOT.
- The DomainManager creates the IDM (Incoming Domain Management) and ODM (Outgoing Domain Management) Event Channels.
- The DomainManager parses the DMD (DomainManager Configuration Descriptor) XML file to obtain the DomainName, DomainManager name, and any services the DomainManager uses. If services are listed that are not yet running, the connection will be placed in a Pending Connections list.
- The DomainManager creates its CORBA object.
- The DomainManager registers with the Naming Service.
- The DomainManager creates a PushConsumer CORBA object which consumes Events.
- The DomainManager connects to the ODM Event Channel as a PushSupplier.

- The DomainManager's PushConsumer registers with the IDM Event Channel.
- The DomainManager creates its Log Port (CF::Port).
- The DomainManager loads previously installed Applications.

4.2.3 RUNNING THE NODE BOOTER (DEVICEMANAGER/SERVER)

4.2.3.1 PRE-CONDITIONS

Before starting the NodeBooter the following pre-condition(s) must be met.

1. In order to run the NodeBooter you must ensure that your environment has been setup properly, as previously described in Section 1.1.1.
2. In order to run the NodeBooter the Naming Service must be running, as previously described in Section 4.2.1.
3. In order to run the NodeBooter the DomainBooter must be running, as previously described in Section 4.2.2.

4.2.3.2 STARTUP

The NodeBooter runs in a terminal window. Open up a new terminal window and from the prompt enter the following command: `startNodeBooter` and hit `<ENTER>`. This will start the NodeBooter executable. This executable may need time to complete its own internal boot up process. Do not move on to the next step until the console window output indicates the executable is running. You should see the output shown in Figure 4-4 to ensure the executable is running. If you receive a message indicating '*Permission Denied*' you may have to change the file permissions for the user to have execute privileges. If you experience other problems, check to make sure that the NodeBooter is not already started. Type "`top`" and hit `<ENTER>` at the command line to view all processes running. If the NodeBooter is listed as a running process, kill the process and start again. Refer to section 3.2.4 Troubleshooting Tips for additional information on changing file permissions or using the "`top`" utility.

4.2.3.3 RUNNING

The output shown in Figure 4-4 should be seen within the console for the NodeBooter. The OrcaCF v1.1.0 (DeviceManager/Server) is now running and accepting messages sent to it by client applications. The output shown in Figure 4-5 should be seen within the console for the DomainBooter. An explanation of the output shown in the figures below will be provided in the next section 4.2.2.4 Explanation. Stopping the OrcaCF v1.1.0 Server applications will be explained in section 4.3 Stopping and Suspending Work.


```
dackerman@coelacanth:~ - Node Booter - Konsole
[dackerman@coelacanth dackerman]$ startNodeBooter

-----
Starting OrcaCF Node Booter
-----

[NODE_BOOT]: TAO ORB has been initialized.
[NODE_BOOT]: RootPOA has been initialized.
[NODE_BOOT]: POA Manager has been initialized and activated.
[NODE_BOOT]: Root Naming Context has been obtained.
[DEVICE_MGR]: LogPort created.
GPPDEV[1]: GPPDeviceEventPort created.
GPPDEV[2]: GPPDevice Registered with DeviceManager.
[DEVICE_MGR]: GPPDevice created.
AUDIODEV[1]: AudioInPort created.
AUDIODEV[2]: AudioOutPort created.
AUDIODEV[3]: AudioEventPort created.
AUDIODEV[4]: AudioDevice Registered with DeviceManager.
[DEVICE_MGR]: AudioDevice created.
[DEVICE_MGR]: AudioDevice initialized.
[AUDIODEV]: Configure: BITS/SAMPLE - 16
[AUDIODEV]: Configure: MONO/STEREO - Stereo
[AUDIODEV]: Configure: SAMPLE RATE(Hz) - 16000
[AUDIODEV]: Configure: BLOCK SIZE - 8192
[AUDIODEV]: Configure: BUFFER SIZE - 32768
[DEVICE_MGR]: AudioDevice configured.

<<<<< LAUNCHING [LogService] >>>>>

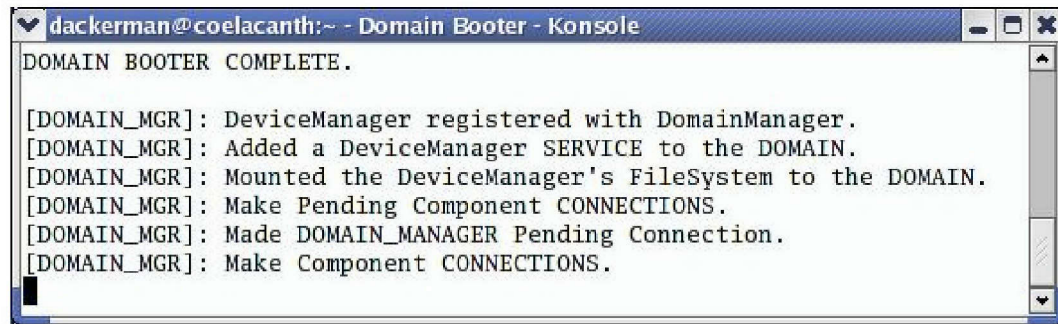
[LOG_SERVICE]: TAO ORB has been initialized.
[LOG_SERVICE]: RootPOA has been initialized.
[LOG_SERVICE]: POA Manager has been initialized and activated.
[LOG_SERVICE]: Registering SERVICE with the DeviceManager...
[DEVICE_MGR]: DeviceManager Configured
[NODE_BOOT]: DeviceManager created.

NODE BOOTER COMPLETE.

[DEVICE_MGR]: Registered SERVICE with the DeviceManager.
[DEVICE_MGR]: DeviceManager Registered with the DomainManager
LOG SERVICE STARTED.
```

Figure 4-4 Example Screenshot of NodeBooter Startup

4.2.3.4



```
dackerman@coelacanth:~ - Domain Booter - Konsole
DOMAIN BOOTER COMPLETE.

[DOMAIN_MGR]: DeviceManager registered with DomainManager.
[DOMAIN_MGR]: Added a DeviceManager SERVICE to the DOMAIN.
[DOMAIN_MGR]: Mounted the DeviceManager's FileSystem to the DOMAIN.
[DOMAIN_MGR]: Make Pending Component CONNECTIONS.
[DOMAIN_MGR]: Made DOMAIN_MANAGER Pending Connection.
[DOMAIN_MGR]: Make Component CONNECTIONS.
```

Figure 4-5 Example Screenshot Showing DomainBooter after NodeBooter Startup

4.2.3.5 EXPLANATION

This section provides an explanation of the steps shown in the figures above. Pre-conditions and a detailed description shall be provided. The pre-conditions are the conditions necessary before execution of the application. The description shall be a detailed explanation of the work being performed during the execution of the application.

Pre-Conditions:

- The Naming Service must be running.
- The Domain Booter must be running.

Description:

- The CORBA environment is initialized first. This includes the initialization of the Object Request Broker (ORB), the Portable Object Adapter (POA), and the POA Manager.
- A reference to the Naming Service is obtained.
- A DeviceManager C++ object is created.
- The DeviceManager creates a CF::FileSystem.
- The DeviceManager creates its CORBA object.
- The DeviceManager creates its Log Port (CF::Port).
- The DeviceManager parses its DCD (Device Configuration Descriptor) XML file to determine what components need to be launched upon boot up.
- The DeviceManager obtains the DomainManager from the Naming Service.
- The DeviceManager launches the components listed in the DCD XML file.
- The DeviceManager registers with the DomainManager, once all the components launched register back with the DeviceManager.
- The DomainManager adds the DeviceManager's Services and Devices to the Domain.
- The DomainManager mounts the DeviceManager's FileSystem to its FileManager.
- The DomainManager makes all connections listed in the DeviceManager's DCD XML file.

4.2.4 RUNNING THE EVENT VIEWER

4.2.4.1 PRE-CONDITIONS

Before starting the Event Viewer the following pre-condition(s) must be met.

1. In order to run the Event Viewer you must ensure that your environment has been setup properly, as previously described in Section 1.1.1.
2. In order to run the Event Viewer the Naming Service must be running, as previously described in Section 4.2.1.
3. In order to run the Event Viewer the DomainBooter must be running, as previously described in Section 4.2.2.

4.2.4.2 STARTUP

The Event Viewer runs in a terminal window. Open up a new terminal window and from the prompt enter the following command: `startEventViewer` and hit `<ENTER>`. This will start the Event Viewer executable. This executable may need time to complete its own internal boot up process. Do not move on to the next step until the console window output indicates the executable is running. You should see the output shown in the Figure 4-6 to ensure the executable is running. If you receive a message indicating *'Permission Denied'* you may have to change the file permissions for the user to have execute privileges. If you experience other problems, check to make sure that the Event Viewer is not already started. Type `"top"` and hit `<ENTER>` at the command line to view all processes running. If the Event Viewer is listed as a running process, kill the process and start again. Refer to section 3.2.4 Troubleshooting Tips for additional information on changing file permissions or using the `"top"` utility.

4.2.4.3 RUNNING

The following output shown in Figure 4-6 should be seen within the console for the Event Viewer. The Event Viewer executable is now running and accepting messages sent to it by client applications. An explanation of the steps shown in the Figure 4-6 will be provided in the next section 4.2.4.4 Explanation. Stopping the Event Viewer application will be explained in section 4.3 Stopping and Suspending Work below.



```
dackerman@coelacanth:~ - Event Viewer - Konsole
[dackerman@coelacanth dackerman]$ startEventViewer

-----
Starting OrcaCF Event Viewer
-----

1. TAO ORB has been initialized.
2. RootPOA has been initialized.
3. POA Manager has been initialized and activated.
4. Root Naming Context has been obtained.
5. DomainManager has been obtained.
6. EventViewer Consumer has been created.
7. EventViewer Consumer registered with the ODM_Channel
8. EventViewer Consumer registered with the IDM_Channel
Running the EventViewer . . .
```

Figure 4-6 Example Screenshot Showing Start of the Event Viewer

4.2.4.4 EXPLANATION

This section provides an explanation of the steps shown in the figure above. Pre-conditions and a detailed description shall be provided. The pre-conditions are the conditions necessary before execution of the application. The description shall be a detailed explanation of the work being performed during the execution of the application.

Pre-Conditions:

- The Naming Service must be running.
- The DomainBooter must be running.

Description:

- The CORBA environment is initialized first. This includes the initialization of the Object Request Broker (ORB), the Portable Object Adapter (POA), and the POA Manager.
- A reference to the Naming Service is obtained.
- A reference to the DomainManager is obtained from the Naming Service.
- Create a PushConsumer CORBA object which consumes Events.
- Register PushConsumer with the ODM Event Channel.
- Register PushConsumer with the IDM Event Channel.

4.2.5 RUNNING THE LOG VIEWER

4.2.5.1 PRE-CONDITIONS

Before starting the Log Viewer the following pre-condition(s) must be met.

1. In order to run the Log Viewer you must ensure that your environment has been setup properly, as previously described in Section 1.1.1.
2. In order to run the Log Viewer the Naming Service must be running, as previously described in Section 4.2.1.
3. In order to run the Log Viewer the DomainBooter must be running, as previously described in Section 4.2.2.
4. In order to run the Log Viewer the NodeBooter must be running, as previously described in Section 4.2.3.

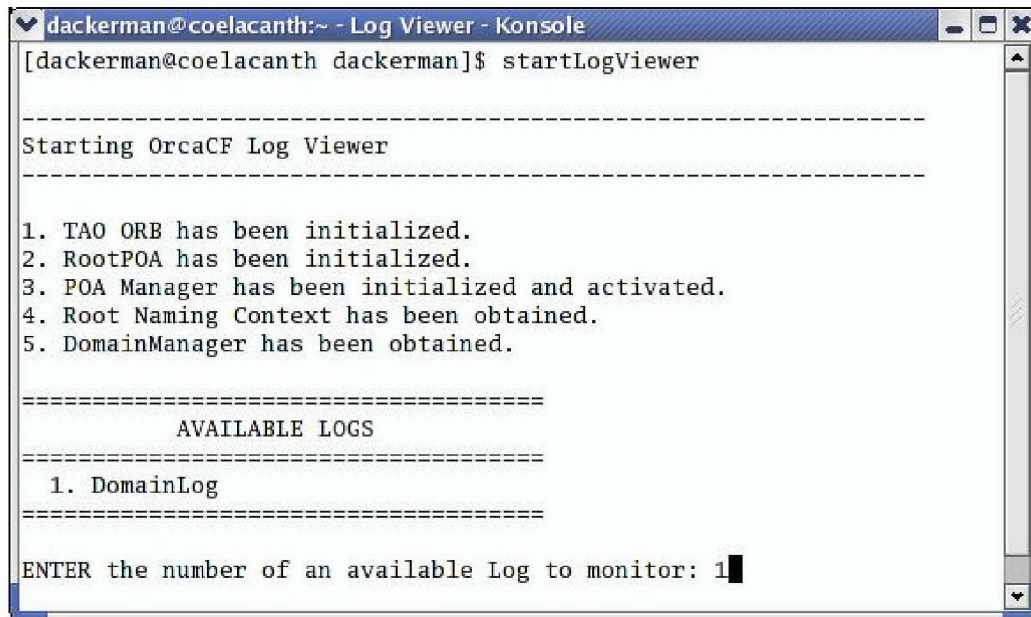
4.2.5.2 STARTUP

The Log Viewer runs in a terminal window. Open up a new terminal window and from the prompt enter the following command: `startLogViewer` and hit `<ENTER>`. This will start the Log Viewer executable. This executable may need time to complete its own internal boot up process. Do not move on to the next step until the console window output indicates the executable is running. You should see the output shown in the Figure 4-7 to ensure the executable is running. If you receive a message indicating *‘Permission Denied’* you may have to change the file permissions for the user to have execute privileges. If you experience other problems, check to make sure that the Log Viewer is not already started. Type `“top”` and hit `<ENTER>` at the command line to view all processes running. If the Log Viewer is listed as a running process, kill the process and start again. Refer to section 3.2.4 Troubleshooting Tips for additional information on changing file permissions or using the `“top”` utility.

4.2.5.3 RUNNING

The output shown in Figure 4-7 should be seen within the console for the Log Viewer. The Log Viewer executable is now running and waiting for a user response. Simply enter `‘1’` to select the [DomainLog](#) and hit `<ENTER>`. You should see the output shown in Figure 4-8. The user may at any time hit the

<SPACEBAR> key to refresh the Log Records displayed in the console window. An explanation of the output shown in the figures below will be provided in the next section 4.2.5.4 Explanation. Stopping the Log Viewer application will be explained in section 4.3 Stopping and Suspending Work below.



```
dackerman@coelacanth:~ - Log Viewer - Konsole
[dackerman@coelacanth dackerman]$ startLogViewer

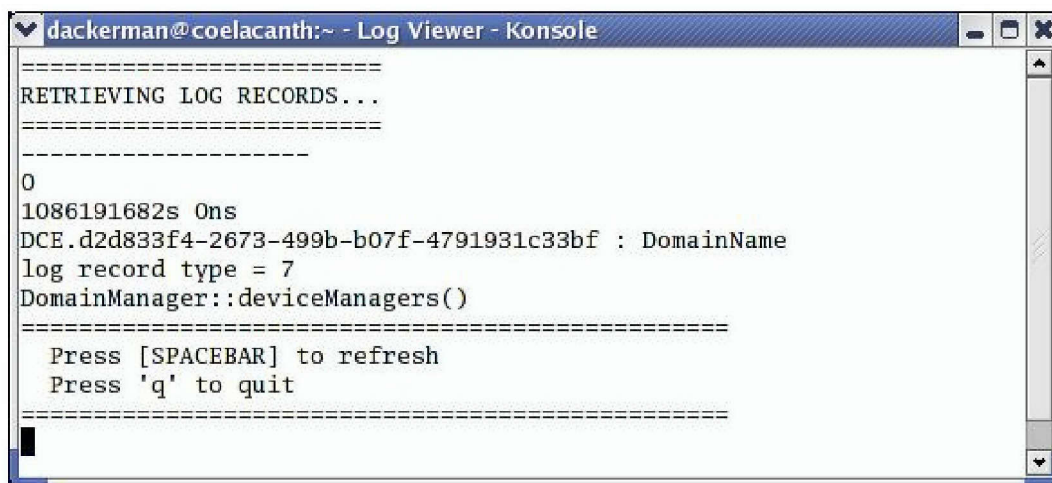
-----
Starting OrcaCF Log Viewer
-----

1. TAO ORB has been initialized.
2. RootPOA has been initialized.
3. POA Manager has been initialized and activated.
4. Root Naming Context has been obtained.
5. DomainManager has been obtained.

=====
                AVAILABLE LOGS
=====
1. DomainLog
=====

ENTER the number of an available Log to monitor: 1
```

Figure 4-7 Example Screenshot Showing Start of the Log Viewer



```
dackerman@coelacanth:~ - Log Viewer - Konsole
=====
RETRIEVING LOG RECORDS...
=====

0
1086191682s Ons
DCE.d2d833f4-2673-499b-b07f-4791931c33bf : DomainName
log record type = 7
DomainManager::deviceManagers()
=====

Press [SPACEBAR] to refresh
Press 'q' to quit
=====
```

Figure 4-8 Example Screenshot Showing Log Viewer after Log selection

4.2.5.4 EXPLANATION

This section provides an explanation of the steps shown in the figures above. Pre-conditions and a detailed description shall be provided. The pre-conditions are the conditions necessary before execution of the application. The description shall be a detailed explanation of the work being performed during the execution of the application.

Pre-Conditions:

- The Naming Service must be running.
- The DomainBooter must be running.
- The NodeBooter must be running.

Description:

- The CORBA environment is initialized first. This includes the initialization of the Object Request Broker (ORB), the Portable Object Adapter (POA), and the POA Manager.
- A reference to the Naming Service is obtained.
- A reference to the DomainManager is obtained from the Naming Service.
- Obtain and list all available Logs in the Domain.
- Get a reference to the Log the user selected.
- Display the Log records when the user presses <SPACEBAR>.

4.2.6 RUNNING THE APPLICATION HMI (HUMAN MACHINE INTERFACE)

4.2.6.1 PRE-CONDITIONS

Before starting the Application HMI the following pre-condition(s) must be met.

1. In order to run the Application HMI you must ensure that your environment has been setup properly, as previously described in Section 1.1.1.
2. In order to run the Application HMI the Naming Service must be running, as previously described in Section 4.2.1.
3. In order to run the Application HMI the DomainBooter must be running, as previously described in Section 4.2.2
4. In order to run the Application HMI the NodeBooter must be running, as previously described in Section 4.2.3

4.2.6.2 STARTUP

The Application HMI runs in a terminal window. Open up a new terminal window and from the prompt enter the following command: `startApplicationHMI` and hit <ENTER>. This will start the Application HMI executable. This executable may need time to complete its own internal boot up process. Do not move on to the next step until the console window output indicates the executable is running. You should see the output shown in the Figure 4-9 to ensure the executable is running. If you receive a message indicating '*Permission Denied*' you may have to change the file permissions for the user to have execute privileges. If you experience other problems, check to make sure that the Application HMI is not already started. Type "`top`" and hit <ENTER> at the command line to view all processes running. If the Application HMI is listed as a running process, kill the process and start again. Refer to section 3.2.4 Troubleshooting Tips for additional information on changing file permissions or using the "`top`" utility.

4.2.6.3 RUNNING

The output shown in Figure 4-9 should be seen within the console for the Application HMI. The Application HMI executable is now running and waiting for a user response.


```
dackerman@coelacanth:~ - Application HMI - Konsole
[dackerman@coelacanth dackerman]$ startApplicationHMI

Starting Application HMI

-----
1. TAO ORB has been initialized.
2. RootPOA has been initialized.
3. POA Manager has been initialized and activated.
4. Root Naming Context has been obtained.
5. DomainManager has been obtained.

=====
AVAILABLE APPLICATION FACTORIES
=====
1. Sound Demo
=====

To CREATE an Application, ENTER the number of an available
Application Factory listed above and press ENTER: █
```

Figure 4-9. Example Screenshot Showing Application HMI Startup

There should be one available *ApplicationFactory* to select from called “[Sound Demo](#)”. This is a Demo Application that acts as a Sound Recorder. Type the number ‘1’ in the console window and hit <ENTER>.

You will then be asked for a name of an *Application* you wish to create. Type in any name (e.g. myApp) and hit <ENTER>. The output shown in Figure 4-10 should be seen within the terminal window for the Application HMI.

```
dackerman@coelacanth:~ - Application HMI - Konsole

=====
AVAILABLE APPLICATION FACTORIES
=====
1. Sound Demo
=====

To CREATE an Application, ENTER the number of an available
Application Factory listed above and press ENTER: 1

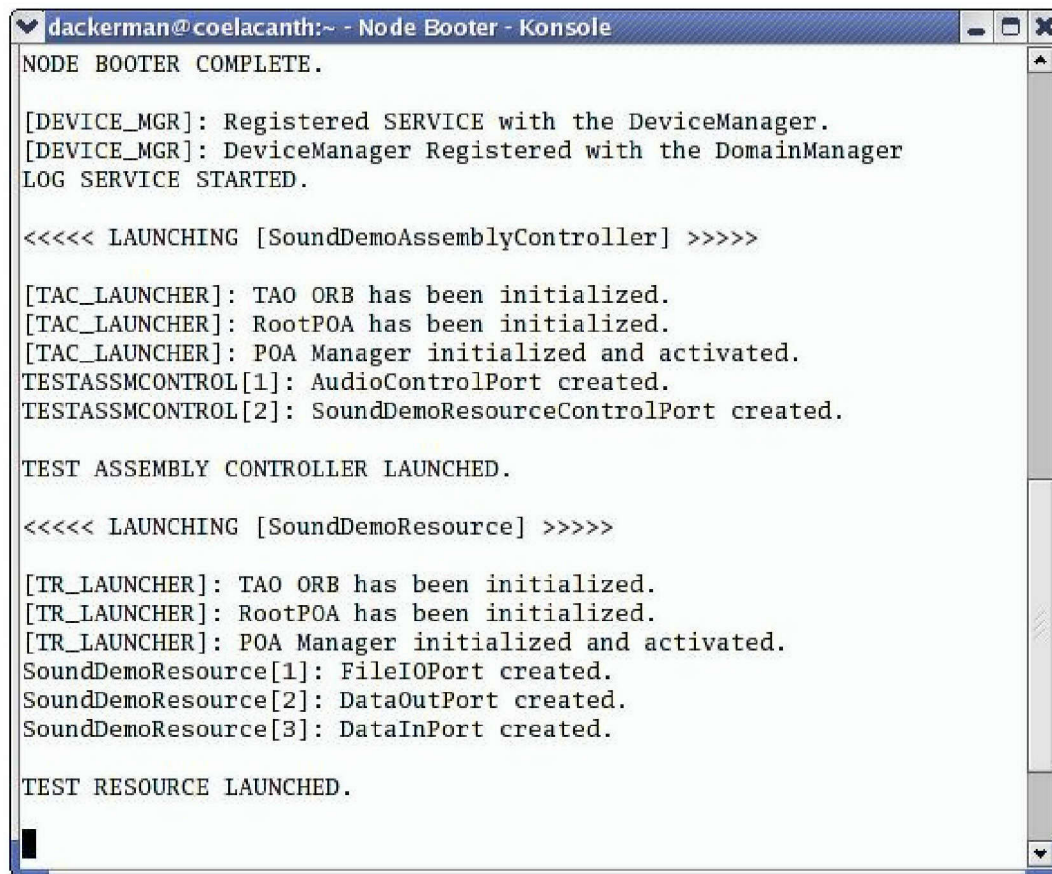
Enter a NAME for the Application you wish to CREATE: myApp

Created Application: myApp

=====
Press 'p' to RUN the Application in PLAY mode.
Press 'r' to RUN the Application in RECORD mode.
Press 's' to STOP the Application.
Press 'q' to QUIT the Application.
=====
█
```

Figure 4-10 Example Screenshot Showing Application HMI After *Application* Creation

The following output shown in Figure 4-11 should now be seen within the terminal window for the NodeBooter.



```
dackerman@coelacanth:~ - Node Booter - Konsole
NODE BOOTER COMPLETE.

[DEVICE_MGR]: Registered SERVICE with the DeviceManager.
[DEVICE_MGR]: DeviceManager Registered with the DomainManager
LOG SERVICE STARTED.

<<<<< LAUNCHING [SoundDemoAssemblyController] >>>>>

[TAC_LAUNCHER]: TAO ORB has been initialized.
[TAC_LAUNCHER]: RootPOA has been initialized.
[TAC_LAUNCHER]: POA Manager initialized and activated.
TESTASSMCONTROL[1]: AudioControlPort created.
TESTASSMCONTROL[2]: SoundDemoResourceControlPort created.

TEST ASSEMBLY CONTROLLER LAUNCHED.

<<<<< LAUNCHING [SoundDemoResource] >>>>>

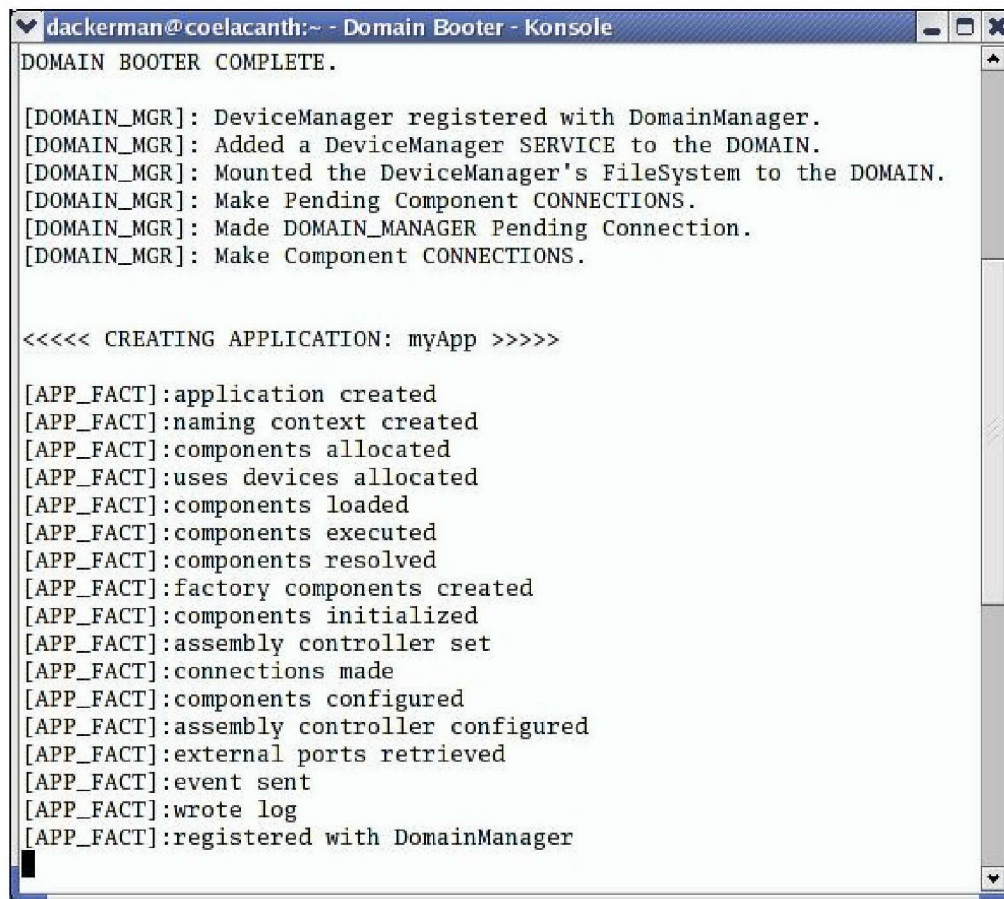
[TR_LAUNCHER]: TAO ORB has been initialized.
[TR_LAUNCHER]: RootPOA has been initialized.
[TR_LAUNCHER]: POA Manager initialized and activated.
SoundDemoResource[1]: FileIOPort created.
SoundDemoResource[2]: DataOutPort created.
SoundDemoResource[3]: DataInPort created.

TEST RESOURCE LAUNCHED.
```

Figure 4-11. Example Screenshot Showing NodeBooter After *Application* Creation

- The Figure 4-11 is the result of selecting an *ApplicationFactory* and creating a new *Application*. Creation of the *Application* initiates the launching of the TestAssemblyController[CF::Resource] and the TestResource[CF::Resource] on the GPPDevice[CF::ExecutableDevice].

The following output shown in Figure 4-12 should be seen within the terminal window for the DomainBooter.



```
dackerman@coelacanth:~ - Domain Booter - Konsole
DOMAIN BOOTER COMPLETE.

[DOMAIN_MGR]: DeviceManager registered with DomainManager.
[DOMAIN_MGR]: Added a DeviceManager SERVICE to the DOMAIN.
[DOMAIN_MGR]: Mounted the DeviceManager's FileSystem to the DOMAIN.
[DOMAIN_MGR]: Make Pending Component CONNECTIONS.
[DOMAIN_MGR]: Made DOMAIN_MANAGER Pending Connection.
[DOMAIN_MGR]: Make Component CONNECTIONS.

<<<<< CREATING APPLICATION: myApp >>>>>

[APP_FACT]:application created
[APP_FACT]:naming context created
[APP_FACT]:components allocated
[APP_FACT]:uses devices allocated
[APP_FACT]:components loaded
[APP_FACT]:components executed
[APP_FACT]:components resolved
[APP_FACT]:factory components created
[APP_FACT]:components initialized
[APP_FACT]:assembly controller set
[APP_FACT]:connections made
[APP_FACT]:components configured
[APP_FACT]:assembly controller configured
[APP_FACT]:external ports retrieved
[APP_FACT]:event sent
[APP_FACT]:wrote log
[APP_FACT]:registered with DomainManager
```

Figure 4-12. Example Screenshot Showing DomainBooter After *Application* Creation

- Figure 4-12 shows the DomainBooter console window after running the `startApplicationHMI` script and creating an *Application*.

The following output shown in Figure 4-13 should be seen within the terminal window for the Event Viewer.



```
dackerman@coelacanth:~ - Event Viewer - Konsole
Running the EventViewer . . .

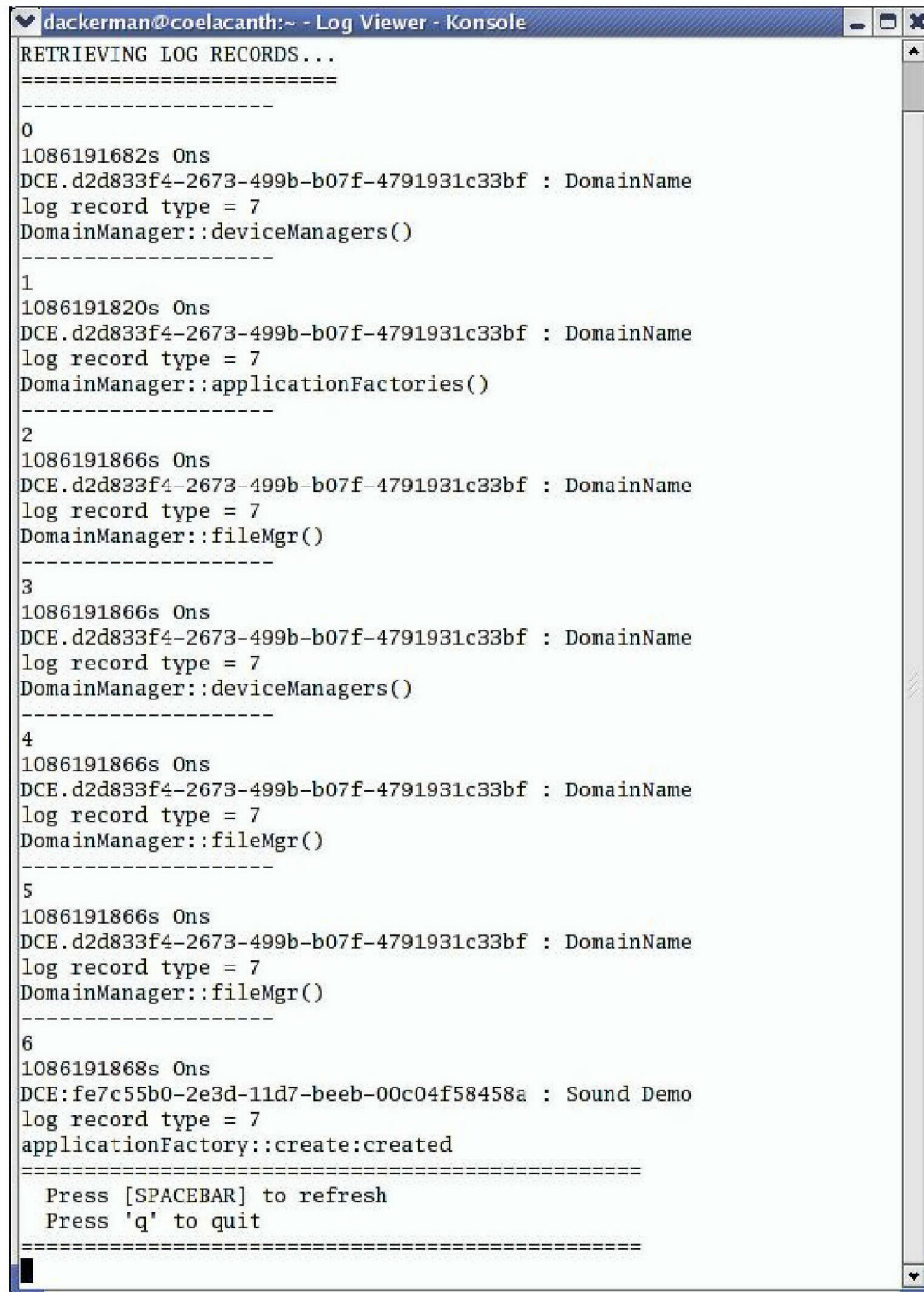
===== ODM_CHANNEL =====

Domain Managment Object Added Event.
  Producer ID: DCE:fe7c55b0-2e3d-11d7-beeb-00c04f58458a
  Source ID: DCE:fe7c55b0-2e3d-11d7-beeb-00c04f58458a:myApp
  Source Name: myApp
  Source Type: Application
=====
```

Figure 4-13. Example Screenshot Showing Event Viewer After *Application* Creation

- Figure 4-13 shows the Events generated from running the `startApplicationHMI` script and creating an *Application*.

At this point you may hit the `<SPACEBAR>` in the Log Viewer console window to view the Log activity at any time. The following output shown in Figure 4-14 shows the Log Viewer console window after user interaction described above.



```

dackerman@coelacanth:~ - Log Viewer - Konsole
RETRIEVING LOG RECORDS...
=====
-----
0
1086191682s Ons
DCE.d2d833f4-2673-499b-b07f-4791931c33bf : DomainName
log record type = 7
DomainManager::deviceManagers()
-----
1
1086191820s Ons
DCE.d2d833f4-2673-499b-b07f-4791931c33bf : DomainName
log record type = 7
DomainManager::applicationFactories()
-----
2
1086191866s Ons
DCE.d2d833f4-2673-499b-b07f-4791931c33bf : DomainName
log record type = 7
DomainManager::fileMgr()
-----
3
1086191866s Ons
DCE.d2d833f4-2673-499b-b07f-4791931c33bf : DomainName
log record type = 7
DomainManager::deviceManagers()
-----
4
1086191866s Ons
DCE.d2d833f4-2673-499b-b07f-4791931c33bf : DomainName
log record type = 7
DomainManager::fileMgr()
-----
5
1086191866s Ons
DCE.d2d833f4-2673-499b-b07f-4791931c33bf : DomainName
log record type = 7
DomainManager::fileMgr()
-----
6
1086191868s Ons
DCE:fe7c55b0-2e3d-11d7-beeb-00c04f58458a : Sound Demo
log record type = 7
applicationFactory::create:created
=====
Press [SPACEBAR] to refresh
Press 'q' to quit
=====

```

Figure 4-14. Example Screenshot Showing Log Activity in the Log Viewer

Before you start recording, you may need to adjust the microphone input volume by using `KMix`, which comes with Red Hat Linux 9.0. Take note that the red light under the microphone is selected, this ensures the microphone is on. This tool may look different depending on which Linux distribution you are running and what sound card you have installed. An example of this tool bar is illustrated in Figure 4-15. `KMix` can be accessed from the following path:

```
KStart -> Sound & Video -> Sound Mixer
```

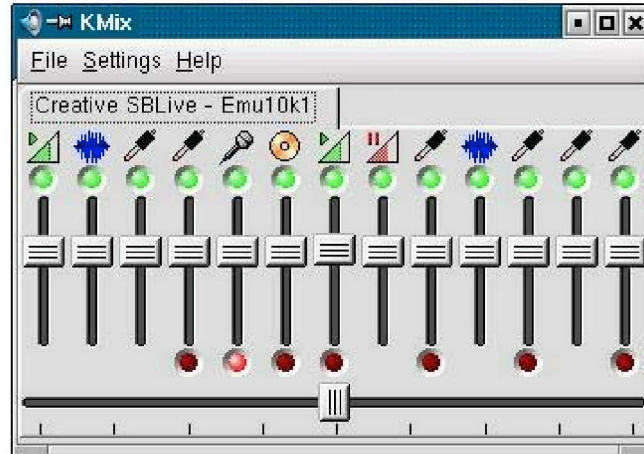


Figure 4-15. `KMix` toolbar for adjusting audio I/O.

- Press the `<r>` key on your keyboard to start the *Application* in **RECORD** mode. You may now talk into the microphone. Your voice will be recorded to a Sound File in the OrcaCF directory.
- Press the `<s>` key to stop recording.
- Press the `<p>` key on your keyboard to start the *Application* in **PLAY** mode. You should be able to hear your voice that you previously recorded. You can adjust the quality and volume by using `Kmix`.
- Press the `<s>` key to stop playing.
- Press the `<q>` key to quit the *Application*.

The output in Figure 4-16 shows the *Application* HMI console window after the user interaction described above.


```
dackerman@coelacanth:~ - Application HMI - Konsole
=====
1. Sound Demo
=====

To CREATE an Application, ENTER the number of an available
Application Factory listed above and press ENTER: 1

Enter a NAME for the Application you wish to CREATE: myApp

Created Application: myApp

=====

Press 'p' to RUN the Application in PLAY mode.
Press 'r' to RUN the Application in RECORD mode.
Press 's' to STOP the Application.
Press 'q' to QUIT the Application.
=====

***{{ RECORDING }}***

***{{ STOPPING }}***

***{{ PLAYING }}***

***{{ STOPPING }}***
Quitting the application...

=== ApplicationHMI TERMINATED ===

[dackerman@coelacanth dackerman]$
```

Figure 4-16. Example Screenshot Showing Application HMI After User Interaction

Note: OrcaCF does not currently allow the user to restart the Application HMI without first terminating all of the other console windows <ctrl><c>. If the user chooses to RECORD some voice input, stop the Application, and begin RECORDING again, the new voice input is appended to the Sound File. PLAYING the Sound File will play all recording sessions performed during the life of the Application.

4.2.6.4 EXPLANATION

This section provides an explanation of the steps shown in the figures above. Pre-conditions and a detailed description shall be provided. The pre-conditions are the conditions necessary before execution of the application. The description shall be a detailed explanation of the work being performed during the execution of the application.

Pre-Conditions:

- The Naming Service must be running.
- The DomainBooter must be running.
- The NodeBooter must be running.

Description:

- The CORBA environment is initialized first. This includes the initialization of the Object Request Broker (ORB), the Portable Object Adapter (POA), and the POA Manager.
- A reference to the Naming Service is obtained.
- A reference to the DomainManager is obtained from the Naming Service.
- Obtain and list all available Application Factories in the Domain.
- The user selects an Application Factory and a name for an Application.
- The Application Factory parses the SAD (Software Assembly Descriptor) XML file to determine what components to allocate, load, and execute.
- The Application Factory makes the connections that are listed in the SAD XML file.
- The Application Factory creates a CF::Application.
- When the user selects <r> the Application gets configured in RECORD mode and starts recording from the microphone.
- When the user selects <s> the Application is stopped.
- When the user selects <p> the Application gets configured in PLAY mode and starts playing the soundfile.
- When the user selects <s> the Application is stopped.
- When the user selects <q> the Application is torn down.

4.3 STOPPING AND SUSPENDING WORK

To conclude your session, you must first shutdown/kill all the OrcaCF processes in the following order:

- Application HMI: When you see the “ApplicationHMI TERMINATED” message, simply close the console window.
- Log Viewer: Shut down the Log Viewer (if it is running). This is accomplished by hitting the <q> key. You should see the prompt and cursor return to the window when the Log Viewer is shut down. Once this is observed, simply close the console window.
- Event Viewer: Shut down the Event Viewer (if it is running). This is accomplished by pressing <ctrl><c> in the console window that you used to start it. You should see the prompt and cursor return to the window when the Event Viewer is shut down. Once this is observed, simply close the console window.
- NodeBooter: Shut down the NodeBooter. This is accomplished by pressing <ctrl><c> in the console window that you used to start it. You should see the prompt and cursor return to the window when the NodeBooter is shut down. Once this is observed, simply close the console window.
- DomainBooter: Shut down the DomainBooter. This is accomplished by pressing <ctrl><c> in the console window that you used to start it. You should see the prompt and cursor return to the window when the DomainBooter is shut down. Once this is observed, simply close the console window.
- Naming Service: Shut down the Naming Service. This is accomplished by pressing <ctrl><c> in the console window that you used to start it. You should see the prompt and cursor return to the window when the Naming Service is shut down. Once this is observed, simply close the console window.

Once the *Application* has been terminated, all terminal windows can be closed. In order to restart the OrcaCF v1.1.0 application, repeat the steps in section 4.2 Initiating a Session.

4.4 UNINSTALLING THE APPLICATION

Uninstalling the OrcaCF v1.1.0 is very straight forward. Simply delete the entire OrcaCF directory and remove entries made in your `.bashrc` file during installation.

5.0 Software Support Information

5.1 “AS BUILT” SOFTWARE DESIGN

Appendix D, Software Design, has been included with this SUM to provide design information regarding the OrcaCF. Appendix D is located in the same directory as this SUM and is named “OrcaCF_SUM_App_D_SoftwareDesign_v1_1_0.pdf”.

6.0 Copyright Notice

Copyright (c) 2004 L-3 Communications Government Services Incorporated (GSI). All rights reserved.

The Open Radio Communications Architecture Core Framework (OrcaCF) is a Core Framework implementation of the Software Communications Architecture (SCA) specification version 2.2. It includes a CORBA ORB, and an XML DOM parser. The SCA Spec is available from <http://jtrs.army.mil/>.

By downloading, installing, using, or modifying this software and/or software documentation, the user agrees to be bound by the terms and conditions of this license, and those licenses of any 3rd party products included in this distribution. This agreement does not limit User's rights under, or grant User rights that supercede, the license terms of any particular component included in this distribution.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License (LGPL) as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version. A copy of the LGPL is distributed with this software, and can also be obtained from the web site <http://www.gnu.org/copyleft/gpl.html>.

UNITED STATES GOVERNMENT RIGHTS: This software was produced for the US Government under Contract No. F30602-01-C-0205, Air Force Research Laboratory (AFRL), Department of the Air Force, and is subject to the Rights in Noncommercial Computer Software and Noncommercial Computer Software Documentation Clause (DFARS) 252.227-7014 (June 1995).

THE LICENSEE AGREES THAT THE US GOVERNMENT WILL NOT BE CHARGED ANY LICENSE FEE AND/OR ROYALTIES RELATED TO EITHER THIS SOFTWARE OR SOFTWARE DOCUMENTATION.

Third Party Licenses:

Some software components used in the development of the OrcaCF are subject to the GNU General Public License such as RED HAT Linux, Fedora Linux, Konqueror, and Kdevelop.

ACE/TAO is made available under the "open source software" model. ACE(TM) and TAO(TM) are copyrighted by Douglas C. Schmidt and his research group at Washington University,

University of California, Irvine, and Vanderbilt University Copyright (c) 1993-2003, all rights reserved. The ACE/TAO license can be found at the following web site: <http://www.cs.wustl.edu/~schmidt/ACE-copying.html>.

This product includes software developed by the Apache Software Foundation <http://www.apache.org/>. The Xerces-C++ XML parser is available in both source distribution and binary distribution. Xerces-C++ is made available under the Apache Software License v1.1. A copy of the Apache Software License can be found at their web site: <http://xml.apache.org/LICENSE>.

Trademarks: "RED HAT" and "FEDORA" are registered trademarks of Red Hat, Inc. "Linux" is a registered trademark of Linus Torvalds. ACE and TAO are registered trademarks of Douglas C. Schmidt and Washington University. All other trademarks are the property of their respective owners.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that they reproduce the above copyright notice, this list of conditions, and the following disclaimer in the documentation, source code, and/or other materials provided with the distribution. See the LPGL for more details on the license terms and conditions.

NO WARRANTY:

JTRS SOFTWARE OR DOCUMENTATION IS PROVIDED "AS IS," WITHOUT WARRANTIES OF ANY KIND, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF DESIGN, MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, OR ARISING FROM A COURSE OF DEALING, USAGE OR TRADE PRACTICE. MOREOVER, JTRS SOFTWARE OR DOCUMENTATION IS PROVIDED WITH NO SUPPORT AND WITHOUT ANY OBLIGATION ON THE PART OF THE U.S. GOVERNMENT, JTRS JPO, JTEL, AFRL, L-3 COMMUNICATIONS GOVERNMENT SERVICES INCORPORATED, SPAWARSSYSCEN CHARLESTON, SPAWARSSYSCEN SAN DIEGO OR ITS EMPLOYEES TO ASSIST IN ITS USE, CORRECTION, MODIFICATION, OR ENHANCEMENT.

THE U.S. GOVERNMENT, JTRS JPO, JTEL, AFRL, L-3 COMMUNICATIONS GOVERNMENT SERVICES INCORPORATED, SPAWARSSYSCEN CHARLESTON, SPAWARSSYSCEN SAN DIEGO AND ITS EMPLOYEES SHALL HAVE NO LIABILITY WITH RESPECT TO THE INFRINGEMENT OF COPYRIGHTS, TRADE SECRETS OR ANY PATENTS BY JTRS SOFTWARE OR DOCUMENTATION OR ANY PART THERE OF. MOREOVER, IN NO EVENT WILL THE U.S. GOVERNMENT, JTRS JPO, JTEL, AFRL, L-3 COMMUNICATIONS GOVERNMENT SERVICES INCORPORATED, SPAWARSSYSCEN CHARLESTON, SPAWARSSYSCEN SAN DIEGO OR ITS EMPLOYEES BE LIABLE FOR ANY LOST REVENUE OR PROFITS OR OTHER SPECIAL, INDIRECT AND CONSEQUENTIAL DAMAGES INCLUDING BUT NOT LIMITED TO PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES, LOSS OF USE, DATA, OR BUSINESS INTERRUPTIONS HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

THE NAMES JTRS SOFTWARE, JTRS DOCUMENTATION, JTRS JPO, JTEL, SPAWARSYSCEN CHARLESTON AND SPAWARSYSCEN SAN DIEGO MAY NOT BE USED TO ENDORSE OR PROMOTE PRODUCTS OR SERVICES DERIVED FROM THIS SOURCE WITHOUT EXPRESS WRITTEN PERMISSION FROM THE PROGRAM DIRECTOR, JTRS JPO. FURTHER, PRODUCTS OR SERVICES DERIVED FROM THIS SOURCE MAY NOT BE CALLED JTRS SOFTWARE, NOR MAY THE NAMES JTRS JPO, JTEL, SPAWARSYSCEN CHARLESTON, OR SPAWARSYSCEN SAN DIEGO APPEAR IN THEIR NAMES, WITHOUT EXPRESS WRITTEN PERMISSION FROM THE PROGRAM DIRECTOR, JTRS JPO.

EXPORT LAWS: THIS LICENSE ADDS NO RESTRICTIONS TO THE EXPORT LAWS OF YOUR JURISDICTION. It is the licensee's responsibility to comply with any export regulations applicable in the licensee's jurisdiction. Under CURRENT (July 2003) U.S. export regulations the following countries are designated U.S. embargoed countries. These include: Burma (Myanmar), Cuba, Iraq, Libya, North Korea, Iran, Syria, Sudan, Zimbabwe, and any other country to which the U.S. has embargoed goods and services.

EXPORT CONTROL: As required by U.S. law, User represents and warrants that it: (a) understands that the Software is subject to export controls under the U.S. Commerce Department's Export Administration Regulations ("EAR"); (b) is not located in a prohibited destination country under the EAR or U.S. sanctions regulations (currently Cuba, Iran, Iraq, Libya, North Korea, Sudan and Syria); (c) will not export, re-export, or transfer the Software to any prohibited destination, entity, or individual without the necessary export license(s) or authorizations(s) from the U.S. Government; (d) will not use or transfer the Software for use in any sensitive nuclear, chemical or biological weapons, or missile technology end-uses unless authorized by the U.S. Government by regulation or specific license; (e) understands and agrees that if it is in the United States and exports or transfers the Software to eligible end users, it will, as required by EAR Section 741.17(e), submit semi-annual reports to the Commerce Department's Bureau of Industry & Security (BIS), which include the name and address (including country) of each transferee; and (f) understands that countries other than the United States may restrict the import, use, or export of encryption products and that it shall be solely responsible for compliance with any such import, use, or export restrictions.

Point of Contact:

For questions regarding support of this software, please send inquiries to OrcaCF.GSI@L-3Com.com

For questions related to the OrcaCF project, contact Michael Gudaitis

Email: mike.gudaitis@L-3Com.com, Phone 315-339-6184

For JTRS questions, refer to <http://jtrs.army.mil>.

7.0 Notes

This section contains notes and general information regarding the OrcaCF v1.1.0 software package.

7.1 LIST OF APPENDICES.

All appendices to this SUM are contained within the same directory as this document.

7.1.1 APPENDIX A: LICENSE INFORMATION

This Appendix to the SUM is intended to inform the user of their legal rights for modifying and redistributing the software and documentation included in the Orca Core Framework Software Package.

Reference: OrcaCF_SUM_App_A_LicenseInformation_v1_1_0_.pdf

7.1.2 APPENDIX B: DEVELOPERS NOTES

This Appendix to the SUM provides general guidance as it relates to programming of the OrcaCF. This document does not cover design or style, but instead provides lessons learned and practical advice for developers to ensure a stable application and consistent source code. Although much of the material covered in the Developers Notes can be reused for other projects, it has been specifically created for the OrcaCF project.

Reference: OrcaCF_SUM_App_B_DevelopersNotes_v1_1_0_.pdf

7.1.3 APPENDIX C: COMPILATION BUILD PROCEDURES

This Appendix to the SUM provides general guidance as it relates to setting up, compiling, and building, the Orca Core Framework (OrcaCF) v1.1.0 project

Reference: OrcaCF_SUM_App_C_CompilationBuildProcedures_v1_1_0_.pdf

7.1.4 APPENDIX D: SOFTWARE DESIGN

This Appendix to the SUM provides an “as-built” design of the Orca Core Framework (CF) v1.1.0. This information gives reviewers an overview of the OrcaCF software design in Unified Modeling Language (UML) format. The design diagrams presented in this document were created with Rational Rose Enterprise Edition.

Reference: OrcaCF_SUM_App_D_SoftwareDesign_v1_1_0_.pdf

7.1.5 APPENDIX E: ERRATA

This Appendix to the SUM explains discrepancies among baselined documentation.

Reference: OrcaCF_SUM_App_E_Errata_v1_1_0_.pdf

7.1.6 APPENDIX F : RELEASE NOTES

This Appendix to the SUM provides notes to developers and users describing changes made since the last release of the LinuxCF software package.

Reference: OrcaCF_SUM_App_F_ReleaseNotes_v1_1_0.pdf

7.1.7 APPENDIX G : CODE STYLE GUIDE

This Appendix to the SUM provides general guidance as it relates to the programming style of the OrcaCF. This document does not cover design or technique, but instead provides a programming style to ensure consistent and readable source code. This Style Guide is used by the L-3 development team as a reference while programming the OrcaCF.

Reference: OrcaCF_SUM_App_G_CodeStyleGuide.pdf

7.2 ACRONYMS

A	
ACE	ADAPTIVE Communication Environment
ADC	Analog-to-Digital Converter
AFRL	Air Force Research Laboratory
ALE	Automatic Link Establishment
AM	Amplitude Modulation
API	Application Program Interface
ASIC	Application Specific Integrated Circuits
ASIP	Advanced System Improvement Program (SINCGARS)
ASN.1	Abstract Syntax Notation 1
ATC	Air Traffic Control
B	
BLOS	Beyond Line Of Sight
bps	Bits per Second
C	
C3I	Command, Control, Communications & Intelligence
C4I	Command, Control, Communications, Computers & Intelligence
CASE	Computer-Aided Software Engineering
CDRL	Contract Data Requirements List
CD-ROM	Compact Disk- Read Only Memory
CF	Core Framework
CI	Configuration Item
CM	Configuration Management
CMM	Capability Maturity Model
CNI	Communications, Navigation and Identification
COM	Computer Operation Manual
CORBA	Common Object Request Broker Architecture standardized by OMG
COTS	Commercial Off-the-Shelf
CP	Change Proposal
cPCI	compact Personal Computer Interface
CPM	Computer Programming Manual
CR	Change Report
CRC	Communications Research Centre
CSCI	Computer Software Configuration Item
D	
DAC	Digital-to-Analog Converter
DAMA	Demand Assigned Multiple Access
DASA	Demand-Assigned Single Access
DBDD	Database Design Description
DCR	Document Change Request
DID	Data Item Description
DII/COE	Defense Information Infrastructure / Common Operating Environment
DoD	Department of Defense
DODI	Department of Defense Instruction
DoN	Department of Navy

DSP	Digital Signal Processor
DTD	Document Type Description
E	
ECCM	Electronic Counter-Counter Measure
ECP	Engineering Change Proposal
EKMS	Electronic Key Management System
EMI	Electromagnetic Interference
EPLRS	Enhanced Position Location Reporting System
F	
FM	Frequency Modulation
FOT	Functional Operability Testing
FPGA	Field Programmable Gate Arrays
FQT	Formal Qualification Test
FSM	Firmware Support Manual
FY	Fiscal Year (Military October to September)
G	
GFI	Government Furnished Information
GHz	Giga Hertz (One Billion Cycles per second)
GP	General Purpose
GPS	Global Positioning System
GUI	Graphical User Interface
H	
HCI	Human-Computer Interface
HMI	Human-Machine Interface
HQ	Have Quick
HW	Hardware
HWCI	Hardware Configuration Item
I	
IA	Information Assurance
IDD	Interface Design Description
IDE	Integrated Development Environment
IDL	Interface Definition Language standardized by OMG
IDM	Incoming Domain Manager
IEEE	Institute of Electronics and Electrical Engineers
IF	Intermediate Frequency
INFOSEC	INFormation SECurity
IOC	Initial Operational Capability
IP	Internet Protocol
IRS	Interface Requirements Specification
IV&V	Independent Verification and Validation
IVT	Interface Validation Tests
J	
JPO	Joint Program Office
JTA	Joint Technical Architecture
JTAP	JTRS Test Application
JTeL	JTRS Technology Laboratory
JTRS	Joint Tactical Radio System

K	
KPA	Key Process Area
L	
LAN	Local Area Network
LCCB	Local Configuration Control Board
OrcaCF	Linux Core Framework
LCM	Life Cycle Maintenance
LOS	Line Of Sight
LPD	Low Probability of Detection
LPE	Low Probability of Exploitation
LPI	Low Probability Of Intercept
M	
MHz	Mega Hertz (One Million Cycles per second)
MLS	Multi-Level Security
MNS	Mission Needs Statement
MSC	Message Sequence Chart
N	
NB	Narrow Band
NSA	National Security Agency
O	
OCD	Operational Concept Description
OE	Operating Environment
ODM	Outgoing Domain Manager
OMG	Object Management Group
OOD	Object-Oriented Design
OPEVAL	Operational Evaluation
ORB	Object Request Broker
OrcaCF	Open Radio Communication Architecture
ORD	Operational Requirements Document
OS	Operating System
OT&E	Operational Test & Evaluation
OTAR	Over The Air Rekey
OTAT	Over The Air Transfer
OTAZ	Over The Air Zeroize
P	
PAL	Process Asset Library
PC	Personal Computer
PDR	Preliminary Design Review
PKI	Public Key Infrastructure
POA	Portable Object Adapter
POSIX	Portable Operating System Interface for UNIX
PP	Program Package
PR	Problem Report
PSM	Practical Software Measurement
PTR	Program Trouble Report
PWF	Pilot Waveform

Q	
QA	Quality Assurance
R	
RF	Radio Frequency
RM	Requirements Management
RMP	Requirements Management Plan
RTOS	Real-Time Operating System
S	
SATCOM	SATellite COMmunications
SATURN	Second Generation Anti-Jam Tactical UHF Radio for NATO. Defined in STANAG 4372.
SCA	Software Communications Architecture
SCCB	System Configuration Control Board
SCM	Software Configuration Management
SCMP	Software Configuration Management Plan
SCOM	Software Center Operator Manual
SDD	Software Design Description
SDF	Software Development File
SDL	Software Development Library
SDP	Software Development Plan
SDR	Software Defined Radio
SDR Forum	Software Defined Radio Forum
SEE	Software Engineering Environment
SEI	Software Engineering Institute
SEM-E	Standard Electronic Module - E (size)
SEN	Software Engineering Notebook
SEPG	Software Engineering Process Group
SEPO	Software Engineering Process Office
SINCGARS	Single Channel Ground and Airborne Radio System
SIOM	Software Input/Output Manual
SIP	Software Installation Plan
SLOC	Source Lines of Code
SOW	Statement of Work
SPE	Software Product Evaluation
SPI	Software Process Improvement
SPIP	Software Process Improvement Plan
SPM	Software Project Manager
SPP	Software Project Plan
SPS	Software Product Specification
SPTO	Software Project Tracking and Oversight
SQA	Software Quality Assurance
SQAP	Software Quality Assurance Plan
SQER	Software Quality Evaluation Report
SQT	System Qualification Test
SRD	Support and Rationale Document
SRS	Software Requirements Specification
SSC-SD	SPAWAR Systems Center San Diego

SSDD	System/Subsystem Design Description
SSS	System/Subsystem Specification
STD	Software Test Description
STE	Software Test Environment
STP	Software Test Plan
STR	Software Trouble Report
STrP	Software Transition Plan
SU	Software Unit
SUM	Software User Manual
SVD	Software Version Description
SW	Software
SYSKOM	System Command
T	
TAO	The ACE ORB
TECHEVAL	Technical Evaluation
TRANSEC	Transmission Security
TR	Test Review
TTCN	Tree and Tabular Combined Notation
U	
UHF	Ultra High Frequency
UML	Universal Modeling Language standardized by OMG
W	
WB	Wide Band
WBS	Work Breakdown Structure
WF	Waveform
WTE	Waveform Test Environment
WTT	Waveform Test Tool
WWW	World Wide Web
X	
XML	eXtensible Markup Language

Appendix A. License Information

SCOPE

This document is intended to inform the user of their legal rights for modifying and redistributing the software and documentation included in the Core Framework Software Package. The license agreement for this product refers you to this file for details concerning terms and conditions applicable to the Core Framework Software Package, third party software code included in this product, and for certain notices and other information L-3 Communications Government Services Inc. must provide to you under its license to certain software code. The relevant terms and conditions, notices, and other information are provided or referenced below. By installing any of the software included with this product, the user agrees to the following terms and conditions.

OVERVIEW

The Open Radio Communication Architecture Core Framework (OrcaCF) Software Package is Copyright © 2004 L-3 Communications Government Services Inc. All rights reserved.

This work was performed under Contract F30602-01-C-0205, Air Force Research Laboratory (AFRL), Department of the Air Force, and is subject to the Rights in Noncommercial Computer Software and Noncommercial Computer Software Documentation Clause (DFARS) 252.227-7014 (June 1995) as shown in Section Error! Reference source not found.. For complete details of the Core Framework license see section Error! Reference source not found. Copyright and Licensing Information for the Core Framework (Documentation) and section Error! Reference source not found. Copyright and Licensing Information for the Core Framework (Software). Users are permitted to copy and distribute verbatim copies of this license, but changing it is not allowed without prior permission from L3.

Some software components used in the development for the Linux machines are subject to the GNU General Public License such as Red Hat Linux, Konqueror, Kdevelop, and Fedora. For complete details on these items, see section.8.1, 0 and 0 respectively. TAO is made available under the "open source software" model. The ACE ORB source code is copyrighted by Dr. Douglas C Schmidt and the Distributed Object (DOC) research group at Washington University in St. Louis. For complete details of the ACE/TAO licensing terms see section 0. You may also visit the ACE/TAO Licensing web site at <http://www.theaceorb.com/product/license.html>.

"This product includes software developed by the Apache Software Foundation (<http://www.apache.org/>).” The Xerces-C++ XML Parser is available in both source distribution and binary distribution. Xerces-C++ is made available under the Apache Software License. A copy of this license is included in section 0. You may also visit the Apache Software License web site at <http://xml.apache.org/LICENSE>.

In addition to Linux, Microsoft Windows NT 4.0 machines were used for development. Microsoft, Windows, Windows 95, Windows NT, and Windows 2000 are either registered trademarks or trademarks

of Microsoft Corporation in the U.S. and/or other countries. All other trademarks are the property of their respective owners.

Licenses

UNITED STATES GOVERNMENT RIGHTS: This software and software documentation was produced for the US Government under Contract No. F30602-01-C-0205, Air Force Research Laboratory, Department of the Air Force, and is subject to the Rights in Noncommercial Computer Software and Noncommercial Computer Software Documentation Clause (DFARS) 252.227-7014 (June 1995).

THE LICENSEE AGREES THAT THE US GOVERNMENT WILL NOT BE CHARGED ANY LICENSE FEE AND/OR ROYALTIES RELATED TO EITHER THIS SOFTWARE OR SOFTWARE DOCUMENTATION.

By downloading, installing, using, or modifying this software and/or software documentation, the user agrees to be bound by the terms and conditions of this license, and those licenses of any 3rd party products included in this distribution. This agreement does not grant User rights that supercede, or limit User rights under, the license terms of any particular component included in the distribution.

EXPORT LAWS: THIS LICENSE ADDS NO RESTRICTIONS TO THE EXPORT LAWS OF YOUR JURISDICTION. It is the licensee's responsibility to comply with any export regulations applicable in the licensee's jurisdiction. Under CURRENT (July 2004) U.S. export regulations the following countries are designated U.S. embargoed countries. These include: Burma (Myanmar), Cuba, Iraq, Libya, North Korea, Iran, Syria, Sudan, Zimbabwe, and any other country to which the U.S. has embargoed goods and services.

GNU LESSER GENERAL PUBLIC LICENSE (LGPL)

Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc.
59 Temple Place - Suite 330, Boston, MA 02111-1307, USA
<http://www.gnu.org/licenses/lgpl.html>

Everyone is permitted to copy and distribute verbatim copies of this license, but changing it is not allowed.

[This is the first released version of the Lesser GPL. It also counts as the successor of the GNU Library Public License, version 2, hence the version number 2.1.]

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages--typically libraries--of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it. For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs. When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library. We call this license the "Lesser" General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library.

A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system. Although the Lesser General Public License is Less protective of the

users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

Terms and Conditions for Copying, Distribution and Modification

1. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you". A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables. The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".) "Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library. Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.
2. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library. You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.
3. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:
 - a) The modified work must itself be a software library.
 - b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.
 - c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.
 - d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the

facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful. (For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.) These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it. Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library. In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

4. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices. Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy. This option is useful when you wish to copy part of the code of the Library into a program that is not a library.
5. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange. If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.
6. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License. However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables. When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law. If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and

small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.) Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

7. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications. You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

- a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

- b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

- c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

- d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

- e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy. For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable. It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

8. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:
 - a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.
 - b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.
9. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.
10. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.
11. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.
12. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library. If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances. It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice. This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

13. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.
14. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.
15. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

16. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.
17. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS (GNU LGPL)

Copyright notice above.

Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111, USA

GNU Free Documentation License (FDL)

Version 1.2, November 2002

Copyright (C) 2000,2001,2002 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307, USA Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

1. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

2. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

3. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 4.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

4. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also

clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

5. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- a. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- b. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- c. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- d. Preserve all the copyright notices of the Document.
- e. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- f. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

- g. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- h. Include an unaltered copy of this License.
- i. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- j. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- k. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- l. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- m. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
- n. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
- o. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

6. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements."

7. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

8. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

9. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

10. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

11. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

END OF TERMS AND CONDITIONS (GNU FDL)

Copyright notice above.

Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111, USA

GNU General Public License (GPL)

Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.

59 Temple Place - Suite 330, Boston, MA 02111-1307, USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

Terms and Conditions for Copying, Distribution and Modification

1. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you". Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.
2. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program. You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.
3. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:
 - a. You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.
 - b. You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

- c. If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it. Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program. In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

- 4. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:
 - a. Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
 - b. Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
 - c. Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

5. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.
6. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.
7. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.
8. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

9. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

10. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.
11. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS (GNU GPL)

Copyright notice above.

Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111, USA

The ACE ORB (TAO)

Introduction

TAO is made available under the "open source software" model. The source is freely downloadable, open for inspection, review and comment. Copies may be freely installed across all your systems and those of your customers. The source code is designed to be compiled and used across a wide variety of hardware and operating systems architectures. Target systems include UNIX systems, including Linux; MS Windows platforms, and real time platforms such as VxWorks, Lynx and pSOS.

The ACE ORB source code is copyrighted by Dr. Douglas C Schmidt and the Distributed Object (DOC) research group at Washington University in St. Louis. The actual terms are given below, are also included in TAO documentation from OCI and the OCI distributions of TAO on CDs. TAO is made available by means of an open source model. TAO may be used without the payment of development license or run time fees. It also means that the TAO source may be made available along, with any added value products, which utilize TAO. The distributions on CD contain the source code for ACE and TAO, in accordance with those requirements.

TAO also includes software from Sun Microsystems. This software is related to the IDL compiler and IIOP. This software also may be freely distributed without fees. The licensing details are included below.

TAO under certain circumstances also uses a software program called GPERF. This software was also written by Dr. Schmidt and is licensed under the terms of the Free Software Foundation's GNU Public License (GPL). Details on this licensing may also be found below.

Please read the licensing information prior to purchase if you have any concerns and to fully understand your obligations as a user.

Copyright and Licensing Information for ACE(TM) and TAO(TM)

The ACE ORB source code is copyrighted by Dr. Douglas C Schmidt and the Distributed Object Computing (DOC) research group at Washington University in St. Louis. The actual terms are reproduced on the CD. TAO is made available by means of an open source model. TAO may be used without the payment of development license or run time fees. TAO source code may be made available along, with any added value products, which utilize TAO. This distribution contains not only the source code for ACE and TAO, in consonance with the spirit of open source practices, but also added value products such as installation scripts, object code for debugging and binaries for commonly used platforms. These are referred to as the software programs. OCI is the authorized distributor of TAO products and services. The use of the ACE, The ACE ORB and TAO trade or service marks is by permission of Washington University.

Warranty

ACE and TAO are provided as is, with no warranties of any kind, including the warranties of design, merchantability and fitness for a particular purpose, non-infringement, or arising from a course of dealing, usage or trade practice. Moreover, ACE and TAO are provided with no support and without any obligation on the part of Washington University, its employees, or students to assist in its use, correction, modification, or enhancement.

Year 2000

Both ACE and TAO have been designed to be Y2K-compliant, as long as the underlying OS platform is Y2K-compliant.

Liability

Washington University, its employees, and students shall have no liability with respect to the infringement of copyrights, trade secrets or any patents by ACE and TAO or any part thereof. Moreover, in no event will Washington University, its employees, or students be liable for any lost revenue or profits or other special, indirect and consequential damages.

Trademarks

The names ACE^(TM), TAO^(TM), and Washington University may not be used to endorse or promote products or services derived from this source without express written permission from Washington University. Further, products or services derived from this source may not be called ACE^(TM) or TAO^(TM), nor may the name Washington University appear in their names, without express written permission from Washington University.

Copyright and Licensing Information for GPERF

GPERF is a standalone software program. GPERF generates perfect hash functions for lookups based on a set of key words when the key words are known in advance. They are called perfect hash functions because only a single access into the data structure is needed in order to perform a lookup. When the set of IDL operations is known in advanced TAO uses the perfect hash functions generated by GPERF in order to perform the operation lookup in constant time. Similarly servant lookups can be done, if the set of servants is known in advanced.

GPERF was originally developed by Professor Douglas Schmidt. Professor Schmidt subsequently signed the copyright over to the Free Software Foundation, causing gperf to be licensed under the GPL (GNU General Public License). The FSF still maintains that version of gperf. When perfect hashing was added as an option to TAO, gperf was selected to provide that function. It was extended and enhanced to meet the more demanding needs of TAO and a derived version was placed in the ACE application libraries. When using TAO under certain circumstances you may elect to use that version of gperf, which is part of the ACE distribution of examples and optional programs. Both the current FSF gperf and the ACE gperf are based on the original implementation. Since the ACE gperf is derived from the original GPL'ed version, it too is licensed under the GPL.

The following terms are found in the source files for gperf:

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

To receive a copy and more information about the GNU General Public License write to the Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA, or visit their web site <http://www.gnu.org> .

COPYRIGHT AND LICENSING INFORMATION FOR IIOP AND THE IDL COMPILER FRONT-END

TAO utilizes software obtained/derived from Sun Microsystems. One product implements the Object Management Group's (OMG) Internet Inter-ORB Protocol. You may copy, modify, distribute, or sublicense the licensed product without charge as part of a product or software program developed by

you, so long as you preserve the functionality of inter-operating with the Object Management Group's "Internet Inter-ORB Protocol" (IIOP) Version one.

A second Sun product implements an OMG Interface Definition Language compiler front end. You may also include this product freely in any distribution, and may modify it, as long as you do not remove functionality.

In both cases you must not use Sun Microsystems name, logo etc. in any subsequent distribution or promotion of your product, and you must include the licensing terms in their entirety.

The detailed Sun licensing terms, are as follows:

Copyright and Licensing for Sun Products

Copyright 1995 Sun Microsystems, Inc.

All Rights Reserved.

This notice applies to all files in this software distribution that were originally derived from SunSoft IIOP code. (Such files contain Sun Microsystems copyright notices).

This software product (LICENSED PRODUCT), implementing the Object Management Group's "Internet Inter-ORB Protocol", is protected by copyright and is distributed under the following license restricting its use. Portions of LICENSED PRODUCT may be protected by one or more U.S. or foreign patents, or pending applications.

LICENSED PRODUCT is made available for your use provided that you include this license and copyright notice on all media and documentation and the software program in which this product is incorporated in whole or part.

You may copy, modify, distribute, or sublicense the LICENSED PRODUCT without charge as part of a product or software program developed by you, so long as you preserve the functionality of inter-operating with the Object Management Group's "Internet Inter-ORB Protocol" version one. However, any uses other than the foregoing uses shall require the express written consent of Sun Microsystems, Inc.

The names of Sun Microsystems, Inc. and any of its subsidiaries or affiliates may not be used in advertising or publicity pertaining to distribution of the LICENSED PRODUCT as permitted herein.

This license is effective until terminated by Sun for failure to comply with this license. Upon termination, you shall destroy or return all code and documentation for the LICENSED PRODUCT.

LICENSED PRODUCT IS PROVIDED AS IS WITH NO WARRANTIES OF ANY KIND INCLUDING THE WARRANTIES OF DESIGN, MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR ARISING FROM A COURSE OF DEALING, USAGE OR TRADE PRACTICE.

LICENSED PRODUCT IS PROVIDED WITH NO SUPPORT AND WITHOUT ANY OBLIGATION ON THE PART OF SUN OR ANY OF ITS SUBSIDIARIES OR AFFILIATES TO ASSIST IN ITS USE, CORRECTION, MODIFICATION OR ENHANCEMENT.

SUN OR ANY OF ITS SUBSIDIARIES OR AFFILIATES SHALL HAVE NO LIABILITY WITH RESPECT TO THE INFRINGEMENT OF COPYRIGHTS, TRADE SECRETS OR ANY PATENTS BY LICENSED PRODUCT OR ANY PART THEREOF.

IN NO EVENT WILL SUN OR ANY OF ITS SUBSIDIARIES OR AFFILIATES BE LIABLE FOR ANY LOST REVENUE OR PROFITS OR OTHER SPECIAL, INDIRECT AND CONSEQUENTIAL DAMAGES, EVEN IF SUN HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

This notice applies to the files used in the IDL Compiler front-end:

Copyright 1992, 1993, 1994 Sun Microsystems, Inc. All Rights Reserved.

This product is protected by copyright and distributed under the following license restricting its use.

The Interface Definition Language Compiler Front End (CFE) is made available for your use provided that you include this license and copyright notice on all media and documentation and the software program in which this product is incorporated in whole or part. You may copy and extend functionality (but may not remove functionality) of the Interface Definition Language CFE without charge, but you are not authorized to license or distribute it to anyone else except as part of a product or program developed by you or with the express written consent of Sun Microsystems, Inc. ("Sun").

The names of Sun Microsystems, Inc. and any of its subsidiaries or affiliates may not be used in advertising or publicity pertaining to distribution of Interface Definition Language CFE as permitted herein. This license is effective until terminated by Sun for failure to comply with this license. Upon termination, you shall destroy or return all code and documentation for the Interface Definition Language CFE.

INTERFACE DEFINITION LANGUAGE CFE IS PROVIDED AS IS WITH NO WARRANTIES OF ANY KIND INCLUDING THE WARRANTIES OF DESIGN, MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, OR ARISING FROM A COURSE OF DEALING, USAGE OR TRADE PRACTICE.

INTERFACE DEFINITION LANGUAGE CFE IS PROVIDED WITH NO SUPPORT AND WITHOUT ANY OBLIGATION ON THE PART OF SUN OR ANY OF ITS SUBSIDIARIES OR AFFILIATES TO ASSIST IN ITS USE, CORRECTION, MODIFICATION OR ENHANCEMENT.

SUN OR ANY OF ITS SUBSIDIARIES OR AFFILIATES SHALL HAVE NO LIABILITY WITH RESPECT TO THE INFRINGEMENT OF COPYRIGHTS, TRADE SECRETS OR ANY PATENTS BY INTERFACE DEFINITION LANGUAGE CFE OR ANY PART THEREOF.

IN NO EVENT WILL SUN OR ANY OF ITS SUBSIDIARIES OR AFFILIATES BE LIABLE FOR ANY LOST REVENUE OR PROFITS OR OTHER SPECIAL, INDIRECT AND CONSEQUENTIAL DAMAGES, EVEN IF SUN HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Both Sun software products are also covered by the following: Use, duplication, or disclosure by the government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 and FAR 52.227-19. SunOS, SunSoft, Sun, Solaris, Sun Microsystems and the Sun logo are trademarks or registered trademarks of Sun Microsystems, Inc.

SunSoft, Inc.
2550 Garcia Avenue
Mountain View, California 94043

Apache Xerces C++ XML Parser

The Apache Software License, Version 1.1
Copyright (c) 1999 The Apache Software Foundation. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The end-user documentation included with the redistribution, if any, must include the following acknowledgment:
"This product includes software developed by the Apache Software Foundation <http://www.apache.org/>." Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear.
4. The names "Xerces" and "Apache Software Foundation" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact apache@apache.org.
5. Products derived from this software may not be called "Apache", nor may "Apache" appear in their name, without prior written permission of the Apache Software Foundation.

THIS SOFTWARE IS PROVIDED ``AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This software consists of voluntary contributions made by many individuals on behalf of the Apache Software Foundation and was originally based on software copyright (c) 1999, International Business Machines, Inc., <http://www.ibm.com/>. For more information on the Apache Software Foundation, please see <http://www.apache.org>.

Red Hat Linux / Konqueror / Kdevelop

General

As used herein, "EULA" means an end user license agreement, and "Software Programs" means, collectively, the Linux Programs, the Third-Party Programs, the Sun Programs, as each of those terms is defined herein.

Red Hat Linux is a modular operating system made up of hundreds of individual software components, each of which was written and copyrighted individually. Each component has its own applicable end user license agreement. Throughout this document the components are referred to, individually and collectively, as the "Linux Programs." Most of the Linux Programs are licensed pursuant to a Linux EULA that permits you to copy, modify, and redistribute the software, in both source code and binary

code forms. However, you must review the on-line documentation that accompanies each of the Linux Programs included in this product for the applicable Linux EULA. Review these Linux EULAs carefully, in order to understand your rights under them and to realize the maximum benefits available to you with Red Hat Linux. Nothing in this license agreement limits your rights under, or grants you rights that supercede, the terms of any applicable Linux EULA.

The "Linux Applications CD - Productivity Edition", and the "Linux Applications CD - Server Edition", include an assortment of applications from third-party vendors. Throughout this document each of these software components are referred to, individually and collectively, as "Third-Party Programs." Generally, each of these Third-Party Programs is licensed to you by the vendor pursuant to an end user license agreement ("Third-Party EULA") that generally permits you to install each of these products on only a single computer for your own individual use. Copying, redistribution, reverse engineering, and/or modification of these components may be prohibited, and you must look to the terms and conditions of the Third-Party EULA to determine your rights and any limitations imposed on you. Any violation by you of the applicable Third-Party EULA terms shall immediately terminate your license under that Third-Party EULA. For the precise terms of the Third-Party EULAs for each of these Third-Party Programs, please check the on-line documentation that accompanies each of them. If you do not agree to abide by the applicable license terms for these Third-Party Programs, then do not install them on your computer. If you wish to install these Third-Party Programs on more than one computer, please contact the vendor of the program to purchase additional licenses.

The Star Office CD includes software licensed to you from Sun Microsystems, Inc., hereinafter, the "Sun Programs." For the precise terms of the license to you for these Sun Programs, please check the on-line documentation that accompanies them or review the Star Office End User Binary Code License posted at www.redhat.com/licenses. If you do not agree to abide by the applicable license terms for these Sun Programs, then do not install them on your computer.

Red Hat Linux itself is a collective work under U.S. copyright law. Subject to the trademark use limitations set forth in this Agreement, Red Hat grants you a license in the collective work pursuant to the GNU General Public License.

Before Installation

CAREFULLY READ THE FOLLOWING TERMS AND CONDITIONS BEFORE INSTALLING ANY OF THE SOFTWARE PROGRAMS. INSTALLING THE SOFTWARE PROGRAMS INDICATES YOUR ACCEPTANCE TO THE TERMS AND CONDITIONS SET FORTH IN THIS DOCUMENT AND OF THE END USER LICENSE AGREEMENT ASSOCIATED WITH THE SOFTWARE PROGRAM. IF YOU DO NOT AGREE WITH THESE TERMS AND CONDITIONS, DO NOT INSTALL THE SOFTWARE PROGRAMS.

THE SOFTWARE PROGRAMS, INCLUDING SOURCE CODE, DOCUMENTATION, APPEARANCE, STRUCTURE AND ORGANIZATION, ARE PROPRIETARY PRODUCTS OF RED HAT, INC. AND OTHERS AND ARE PROTECTED BY COPYRIGHT AND OTHER LAWS. TITLE TO THESE PROGRAMS, OR TO ANY COPY, MODIFICATION OR MERGED PORTION OF ANY OF THESE PROGRAMS, SHALL AT ALL TIMES REMAIN WITH THE AFOREMENTIONED, SUBJECT TO THE TERMS AND CONDITIONS OF THE APPLICABLE EULA RELATED TO THE SOFTWARE PROGRAMS UNDER CONSIDERATION.

THE "RED HAT" TRADEMARK AND RED HAT'S SHADOW MAN LOGO ARE REGISTERED TRADEMARKS OF RED HAT, INC. IN THE UNITED STATES AND OTHER COUNTRIES. WHILE THIS LICENSE AGREEMENT ALLOWS YOU TO COPY, MODIFY AND DISTRIBUTE THE SOFTWARE, IT DOES NOT PERMIT YOU TO DISTRIBUTE THE SOFTWARE UTILIZING RED

HAT'S TRADEMARKS. YOU SHOULD READ THE INFORMATION FOUND AT http://www.redhat.com/about/trademark_guidelines.html BEFORE DISTRIBUTING A COPY OF THE SOFTWARE, REGARDLESS OF WHETHER IT HAS BEEN MODIFIED.

CERTAIN LIMITED TECHNICAL SUPPORT SERVICES ACCOMPANY RED HAT LINUX. THE RIGHT TO USE THOSE TECHNICAL SUPPORT SERVICES ARE LIMITED TO THE ORIGINAL PURCHASE OF THE PRODUCT FROM EITHER RED HAT OR A RED HAT AUTHORIZED DISTRIBUTOR. WHILE YOU HAVE THE RIGHT TO TRANSFER YOUR COPY OF RED HAT LINUX TO ANOTHER PARTY, YOU MAY NOT TRANSFER THE RIGHT TO USE THOSE TECHNICAL SUPPORT SERVICES ONCE YOU HAVE ACTIVATED YOUR PRODUCT FOR SUPPORT. ANY ATTEMPT TO TRANSFER TECHNICAL SUPPORT SERVICES FOLLOWING ACTIVATION WILL RENDER YOUR RIGHT TO THE TECHNICAL SUPPORT SERVICES NULL AND VOID.

Limited Warranty

EXCEPT AS SPECIFICALLY STATED IN THIS AGREEMENT OR IN AN EULA, THE SOFTWARE PROGRAMS ARE PROVIDED AND LICENSED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

Red Hat, Inc. warrants that the media on which any of the Software Programs are furnished will be free from defects in materials and manufacture under normal use for a period of 30 days from the date of delivery to you. Red Hat, Inc. does not warrant that the functions contained in the Software Programs will meet your requirements or that the operation of the Software Programs will be entirely error free or appear precisely as described in the accompanying documentation.

ANY WARRANTY OR REMEDY PROVIDED UNDER THIS AGREEMENT EXTENDS ONLY TO THE PARTY WHO PURCHASES RED HAT LINUX FROM RED HAT OR A RED HAT AUTHORIZED DISTRIBUTOR.

Limitation of Remedies and Liability

To the maximum extent permitted by applicable law, the remedies described below are accepted by you as your only remedies, and shall be available to you only if you or your dealer registers this product with Red Hat, Inc. in accordance with the instructions provided with this product within ten days after delivery of the Software Programs to you.

Red Hat, Inc.'s entire liability, and your exclusive remedies, shall be: if the Software Programs media are defective, you may return them within 30 days of delivery to you along with a copy of your receipt and Red Hat, Inc., at its option, will replace them or refund the money paid by you for the Software Programs. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, IN NO EVENT WILL RED HAT, INC. BE LIABLE TO YOU FOR ANY DAMAGES, INCLUDING LOST PROFITS, LOST SAVINGS, OR OTHER INCIDENTAL OR CONSEQUENTIAL DAMAGES, ARISING OUT OF THE USE OR INABILITY TO USE THE SOFTWARE PROGRAMS, EVEN IF RED HAT, INC. OR A DEALER AUTHORIZED BY RED HAT, INC. HAD BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

GENERAL

If any provision of this Agreement is held to be unenforceable, that shall not effect the enforceability of the remaining provisions. This Agreement shall be governed by the laws of the State of North Carolina and of the United States, without regard to any conflict of laws provisions.

Copyright © 2002 Red Hat, Inc. All rights reserved. "Red Hat" and the Red Hat "Shadow Man" logo are registered trademarks of Red Hat, Inc. "Linux" is a registered trademark of Linus Torvalds. All other trademarks are the property of their respective owners.

FEDORA TM License Agreement

This agreement governs the download, installation or use of the Software (as defined below) and any updates to the Software, regardless of the delivery mechanism. The Software is a collective work under U.S. Copyright Law. Subject to the following terms, Fedora Project grants to the user ("User") a license to this collective work pursuant to the GNU General Public License. By downloading, installing or using the Software, User agrees to the terms of this agreement.

1. The Software. Fedora Core (the "Software") is a modular Linux operating system consisting of hundreds of software components. The end user license agreement for each component is located in the component's source code. With the exception of certain image files containing the Fedora trademark identified in Section 2 below, the license terms for the components permit User to copy, modify, and redistribute the component, in both source code and binary code forms. This agreement does not limit User's rights under, or grant User rights that supersede, the license terms of any particular component.
2. Intellectual Property Rights. The Software and each of its components, including the source code, documentation, appearance, structure and organization are copyrighted by Fedora Project and others and are protected under copyright and other laws. Title to the Software and any component, or to any copy, modification, or merged portion shall remain with the aforementioned, subject to the applicable license. The "Fedora" trademark is a trademark of Red Hat, Inc. ("Red Hat") in the U.S. and other countries and is used by permission. This agreement permits User to distribute unmodified copies of Software using the Fedora trademark on the condition that User follows Red Hat's trademark guidelines located at <http://fedora.redhat.com/legal/>. User must abide by these trademark guidelines when distributing the Software, regardless of whether the Software has been modified. If User modifies the Software, then User must replace all images containing the "Fedora" trademark. Those images are found in the anaconda-images and the fedora-logos packages. Merely deleting these files may corrupt the Software.
3. Limited Warranty. Except as specifically stated in this agreement or a license for a particular component, to the maximum extent permitted under applicable law, the Software and the components are provided and licensed "as is" without warranty of any kind, expressed or implied, including the implied warranties of merchantability, non-infringement or fitness for a particular purpose. Neither the Fedora Project nor Red Hat warrants that the functions contained in the Software will meet User's requirements or that the operation of the Software will be entirely error free or appear precisely as described in the accompanying documentation. Use of the Software is at User's own risk.
4. Limitation of Remedies and Liability. To the maximum extent permitted by applicable law, Fedora Project and Red Hat will not be liable to User for any damages, including incidental or

consequential damages, lost profits or lost savings arising out of the use or inability to use the Software, even if Fedora Project or Red Hat has been advised of the possibility of such damages.

5. Export Control. As required by U.S. law, User represents and warrants that it: (a) understands that the Software is subject to export controls under the U.S. Commerce Department's Export Administration Regulations ("EAR"); (b) is not located in a prohibited destination country under the EAR or U.S. sanctions regulations (currently Cuba, Iran, Iraq, Libya, North Korea, Sudan and Syria); (c) will not export, re-export, or transfer the Software to any prohibited destination, entity, or individual without the necessary export license(s) or authorizations(s) from the U.S. Government; (d) will not use or transfer the Software for use in any sensitive nuclear, chemical or biological weapons, or missile technology end-uses unless authorized by the U.S. Government by regulation or specific license; (e) understands and agrees that if it is in the United States and exports or transfers the Software to eligible end users, it will, as required by EAR Section 741.17(e), submit semi-annual reports to the Commerce Department's Bureau of Industry & Security (BIS), which include the name and address (including country) of each transferee; and (f) understands that countries other than the United States may restrict the import, use, or export of encryption products and that it shall be solely responsible for compliance with any such import, use, or export restrictions.
6. General. If any provision of this agreement is held to be unenforceable, that shall not affect the enforceability of the remaining provisions. This agreement shall be governed by the laws of the State of North Carolina and of the United States, without regard to any conflict of laws provisions, except that the United Nations Convention on the International Sale of Goods shall not apply.
7. Copyright © 2003 Fedora Project. All rights reserved. "Red Hat" and "Fedora" are trademarks of Red Hat, Inc. "Linux" is a registered trademark of Linus Torvalds. All other trademarks are the property of their respective owners. For more information, see <http://fedora.redhat.com/licenses>.

Appendix B. Developers Notes

Scope

This document describes notes to developers, including guidelines and best practices used for the development of the Open Radio Communications Architecture Core Framework (OrcaCF). These were distilled from engineering notes and design decisions made during the development. Much of the material in this document is specific to the OrcaCF development, which utilizes ANSI C++, CORBA, ACE/TAO and Linux.

Overview

This Appendix to the Software Users Manual (SUM) provides general guidance as it relates to programming of the OrcaCF. This document does not cover design or style, but instead provides lessons learned and practical advice for developers to ensure a stable application and consistent source code. Although much of the material covered in the Developers' Notes can be reused for other projects, it has been specifically created for the OrcaCF project.

The Developers Notes are used by the OrcaCF development team as a reference while implementing the OrcaCF. It currently covers the development resources utilized by the OrcaCF. The remaining sections of this document are organized in the following manner.

Section 0 is a list of reference documents.

Section 10.0 provides ANSI C++ and Standard Template Library (STL) programming guidelines.

Section 0 provides CORBA specific programming guidelines.

Section 0 provides guidelines that are specific to ACE/TAO.

Section 0 provides reference material for working with XML.

Relationship to Other Documents

The Developers Notes provide a standard set of rules for coding the OrcaCF that goes beyond the basic programming styles covered in the Code Style Guide. The intent of appending this document to the SUM is to provide lessons learned by the OrcaCF development team and to aid in the review of the application.

Referenced Documents

- a. Redhat Linux website: <http://www.redhat.com>
- b. ACE/TAO websites: <http://www.cs.wustl.edu/~schmidt/TAO.html>, <http://www.theaceorb.com/>
- c. OCI TAO Developers Guide version 1.2a, volume 1&2 (Part numbers 510-01, 510-02), Object Computing Inc., 2002.
- d. Pure CORBA by Fintan Bolton, 5th Edition, ISBN 0-672-321812-1. Sam's Publishing, 2002.
- e. Advanced CORBA Programming with C++, by Michi Henning & Steve Vinoski , ISBN 0-201-37927-9. Addison-Wesley 1999.
- f. ANSI String Class by Dr. Mark J. Sebern, version 1.7, 12/14/1999
<http://www.msoe.edu/eecs/cese/resources/stl/string.htm>
- g. STL Vector Class by Dr. Mark J. Sebern, version 1.5, 12/14/1999
<http://www.msoe.edu/eecs/cese/resources/stl/vector.htm>
- h. The Waite Groups' Object-Oriented Programming in C++, Third Edition, by Robert LaFore: 1999.

ANSI C++ & STL

Memory Management

Due to the structure of the OrcaCF, it is essential that memory management be handled carefully and thoroughly. Improper memory management can quickly lead to an unstable OrcaCF implementation. This section addresses the creation, allocation, deallocation and destruction of C++ objects used by the OrcaCF.

Object Creation

Any C++ objects that are created must be released. If an object is created explicitly within a function or class, it will be automatically deallocated when it goes out of scope. If a C++ object is allocated implicitly with a pointer reference, the object must be deallocated before the pointer goes out of scope.

All objects should be allocated using the 'new' directive. By using the 'new' directive, objects are allocated on the heap, without the 'new' directive, objects are allocated on the stack. The heap is a more stable memory space for objects than the stack.

```
File myFile = new File();      // created explicitly
File * pMyFile = new File();   // created implicitly
```

Object Destruction

If a C++ object is created explicitly, it will automatically be cleaned up when it goes out of scope. If a C++ object is allocated implicitly with a pointer reference, it must be cleaned up in one of the following manners:

- a. Call an object function that releases the object. (i.e. `fclose()` on a `File` object).
- j. Use the *delete operator* of the object if it was created on the heap. (i.e. created using the 'new' directive).
- k. Use the *destroy operator* if the object was not created on the heap. This will call the object's `destructor()` function.
- l. Pass ownership of the object to CORBA using the `poa->activate_object()` function call.

Whenever an object is deallocated, the pointer should first be checked for NULL to ensure the object was not previously deallocated. If the pointer is not NULL, the object should be deallocated and the pointer set to NULL as shown in the example below.

```
Example:
if (pMyFile != NULL)
{
    delete pMyFile;
    pMyFile = NULL;
}
```

The above code snippet should be included at the end of any function, or within any exception blocks where a C++ object has been allocated. The above code snippet should also be included within a class `destructor()` if there are any class member variables that are C++ objects.

Strings

During the development of the OrcaCF, memory allocation/deallocation problems ("Memory Leaks") were encountered. Debugging and tracking of these memory leaks were performed, the cause was

determined to be a manipulation of strings. The initial intention of the OrcaCF File Service design was to utilize the CORBA::String_var type and ACE_OS string manipulation functions for portability. Research was performed and a design decision was made to utilize the std::string class from the C++ Standard Template Library. Implementation of the std::string class cleared up the memory leaks and was carried through all existing code and will be utilized throughout the rest of the development. For information on how to use the std::string class, refer to section 0.0 above.

An additional note concerning the use of the std::string class pertains to the use of the c_str() operation for returning *const char ** data from a function call. This should never be done because the string is deallocated upon exit of the function and the data can be corrupted by the time the calling function receives the data. In order to avoid this problem, the std::string value should be copied into a CORBA::String_var and the .retn() function of the CORBA::String_var utilized within the return statement. An example of this can be seen below.

```
string_var = CORBA::string_dup(string.c_str());  
return string_var._retn();
```

Sequential Containers

During implementation of the OrcaCF, it became necessary to track and maintain sequences of CORBA Objects that were not defined within the IDL. It was then decided to utilize the STL Sequential Container classes (e.g. Vector, Deque, etc.) to maintain sequences of CORBA Objects.

Upon implementation of the OrcaCF using these container classes, it was discovered that references to CORBA Objects were not being properly managed. In order to manage CORBA references properly the following rules need to be applied when using STL container classes with CORBA references.

2. CORBA references stored within container classes should be VAR types. This ensures that when the container goes out of scope, or when the container element is removed, the CORBA Object is properly cleaned up.
3. A CORBA reference or a structure containing a CORBA reference should never be pushed (e.g. push_back(), push_front()) into a STL container. Instead, an empty element should be pushed into the container. The element within the container should then be referenced and populated appropriately.

Here is some example code demonstrating the above process.

```
std::vector myvect<CORBA::String_var>;  
CORBA::String_var emptyString = CORBA::string_dup("");  
CORBA::String_var myString1 = CORBA::string_dup("my string 1");  
CORBA::String_var myString2 = CORBA::string_dup("my string 2");  
  
myvect.push_back(emptyString);  
myvect[0] = myString1;  
myvect.push_back(emptyString);  
myvect[1] = myString2;
```

CORBA

Portable Object Adapter (POA)

For the design of each object of the OrcaCF, the POA should be studied and a set of policies determined. Table 10-1 lists the policies for a POA, along with the defaults for new child POAs and for the Root

POA. In most cases, the Root POA will be utilized, but these policies must be reviewed to ensure proper implementation of a core framework object.

Policy	Options	Default
LifespanPolicy	TRANSIENT PERSISTENT	TRANSIENT
IdAssignmentPolicy	USER_ID SYSTEM_ID	SYSTEM_ID
IdUniquenessPolicy	UNIQUE_ID MULTIPLE_ID	UNIQUE_ID
ImplicitActivationPolicy	IMPLICIT_ACTIVATION NO_IMPLICIT_ACTIVATION	root=IMPLICIT_ACTIVATION NO_IMPLICIT_ACTIVATION
RequestProcessingPolicy	USE_ACTIVE_OBJECT_MAP_ON LY USE_DEFAULT_SERVANT USE_SERVANT_MANAGER	USE_ACTIVE_OBJECT_MAP_ONL Y
ServantRetentionPolicy	RETAIN NON_RETAIN	RETAIN
ThreadPolicy	ORB_CTRL_MODEL SINGLE_THREAD_MODEL	ORB_CTRL_MODEL

Table 10-1 POA policies

Object Creation

Implicit activation of CORBA objects, such as activation of CORBA objects using the `_this()` function call, should not be used. According to Pure CORBA (p133-134), it is not always readily apparent what POA is used in the creation of a CORBA object when using the `_this()` function call. Also, assuming that we will always be creating under the rootPOA is a flexibility limitation of our Servants. Instead, we should be doing the following:

- Pass in a POA reference through the constructor to any Servant Implementation that will be creating CORBA Objects; i.e. `FileSystem` and `DeviceManager`.
- Save a local copy of the POA passed to the Servant as `mPOA_var`. This will be the POA that will be used whenever that Servant creates a CORBA object. We also need to ensure that the POA reference is destroyed when the implementation object is destroyed.
- We should use the explicit CORBA object creation process utilizing the local POA. The following sample code will accomplish this.

Example:

```
CF_File_i* pFile = NULL;
CF::File_var file_var = CF::File::_nil();
PortableServer::ObjectId_var fileObjId_var;
CORBA::Object_var fileObj_var;
...
```

```

pFile = new CF_File_i(...);
fileObjId_var = mPOA_var->activate_object(pFile, ...);
fileObj_var = mPOA_var->id_to_reference(fileObjId_var.in(), ...);
file_var = CF::File::_narrow(fileObj_var);
...
return file_var._retn();

```

Using the above for the creation of CORBA objects will ensure that we know what POA is used to control the object. It also gives us added flexibility to utilize a POA other than the rootPOA to control the life cycle of an object, along with other POA settings, without explicitly changing the code within the class.

Parameter Passing

Just about all CORBA objects are passed as a reference in one form or another. The danger in passing references is keeping track of ownership and cleanup duties. One of the ways that CORBA helps in keeping track of these items is by providing parameter delimiters. These parameter delimiters are `in`, `inout`, `out` and `_retn`. Each one of these will be discussed below in more detail, providing the responsibilities and some items to watch out for.

`.in()`

allocation – caller: It is the responsibility of the calling function to allocate this object before passing it into a function. The caller should use the `.in()` designator on the object's `_var` type.

initialization – caller: It is the responsibility of the calling function to ensure that the object has been initialized. The caller should use the `.in()` designator on the object's `_var` type.

deallocation – caller: It is the responsibility of the calling function to deallocate the object. The function to which this parameter is passed should never deallocate or take ownership of this parameter (i.e. never set a `.in()` parameter equal to a `_var` type). If a persistent copy of a parameter is needed, the `_duplicate()` function or the `string_dup()` function must be used, and then the callee will be responsible for releasing that persistent copy.

`.inout()`

allocation – caller: This needs to be dynamically allocated (`string_dup`, `_ptr` or `_var`) by the caller. For the OrcaCF, a `_var` type should always be used with the `.inout()` designator.

initialization – caller: It is the responsibility of the calling function to ensure that the object has been initialized. The caller should use the `.inout()` designator on the object's `_var` type.

deallocation – caller: It is the responsibility of the calling function to deallocate the object. The function to which this parameter is passed is responsible for ensuring that an allocated and initialized parameter is passed back to the caller. The callee may deallocate, reallocate and re-initialize the parameter, which is why the parameter must be dynamically allocated.

`.out()`

allocation – proxy: The proxy object is responsible for the allocation of the parameter. The caller should not perform this action when using a `.out()` parameter in a function.

initialization – callee: It is the responsibility of the function called to initialize this parameter.

deallocation – caller: Once a .out() parameter has been returned to the calling function, it is the responsibility of that function to deallocate the parameter.

`._retn()`

allocation – proxy: Much like the .out() parameter, a return value is allocated by the proxy object.

initialization – callee: The function being called is responsible for initialization of the value to be returned. A var type with the .retn() directive should always be used.

deallocation – caller: Once the calling function has received the return value, it is the responsibility of the caller to ensure that the return value is deallocated.

Memory Management

CORBA manages memory allocated to objects by maintaining Reference Counts (RC) to the objects. When the Reference Count reaches 0, the object is deallocated.

CORBA uses three different types for objects, the actual object, which is delineated by a `_obj`, a simple pointer, which is delineated by a `_ptr`, and a smart pointer, which is delineated by a `_var`. Each declaration of the object, regardless of CORBA type, will result in the object having an initial Reference Count of 1. Declaration of each of these object types can be seen below:

```
CF::File myFile_obj = new CF::File();      // CORBA object
CF::File * myFile_ptr = new CF::File();    // CORBA simple pointer
CF::File_ptr myFile_ptr = new CF::File();  // CORBA simple pointer
CF::File_var myFile_var = new CF::File();  // CORBA smart pointer
```

The object type is rarely used because the idea behind CORBA is to pass references to objects that are distributed over a system or network. Simple pointer types reference an object, but do not provide any memory management on their own. When using simple pointers, objects must be explicitly destroyed using the `CORBA::release()` function.

Smart pointers, or `_var` types, are the preferred method of memory management within a CORBA application. Smart pointers increment the Reference Counts when they are created, and they decrement the Reference Counts when they are destroyed or go out of scope. When all smart pointers go out of scope, the Reference Count drops to 0 and the object is destroyed.

Regardless of the CORBA type used to reference objects, great care and understanding must be used when implementing object references, or using the `release()` or `duplicate()` functions on object references. Table 10-2 shows a summary of some of the assignment operations and their effect on the object's Reference Count (RC).

Assignment	Effect
<code>my_ptr = your_ptr</code>	No effect on RC.
<code>My_ptr = your_var</code>	Does not increase RC, but RC will be decremented by 1 when <code>your_var</code> goes out of scope.
<code>My_var = your_ptr</code>	Does not change the RC, but RC will still be decremented by 1 when <code>my_var</code> goes out of scope.
<code>My_var = your_var</code>	The RC for <code>my_var</code> will be decremented by 1. The RC for <code>your_var</code> will be increased by 1, the RC will be decremented by 1 when each <code>_var</code> goes out of scope.
<code>My_ptr = duplicate(your_ptr)</code>	RC is increased by 1, <code>release()</code> must be called on each pointer in order to deallocate the object.
<code>My_var = duplicate(my_ptr)</code>	RC is increased by 1, and will be decremented by 1 when <code>my_var</code> goes out of scope.
<code>My_var = duplicate(your_var)</code>	UNSURE OF RESULT TO RC, this type of assignment should only be used when widening an object reference.
<code>Release(my_ptr)</code>	Decrements RC by 1.
<code>RELEASE(MY_VAR)</code>	DO NOT USE, <code>my_var</code> will call <code>release()</code> on its own when it goes out of scope.

Table 10-2 CORBA Assignment Operations

Releasing CORBA Objects

In some cases, a program cannot wait until all of the references to a CORBA Object go out of scope in order to release itself from the CORBA Environment. Within the SCA there are some particular requirements where this applies, such as the `CF::Resource.releaseObject` operation or the `CF::File.close` operation. In order to safely release a CORBA Object, there are several steps that must be accomplished.

4. The servant class for the CORBA Object must inherit the *public virtual PortableServer::RefCountServantBase*.
5. The POA must be passed into the servant class constructor and the servant class should activate itself with the following code.

```
// activate object
objectId_var = mPOA_var->activate_object(this, ACE_TRY_ENV);
ACE_TRY_CHECK;
object_var = mPOA_var->id_to_reference(objectId_var.in(), ACE_TRY_ENV);
ACE_TRY_CHECK;
mFile_var = CF::File::_narrow(object_var.in(), ACE_TRY_ENV);
ACE_TRY_CHECK;

// verify reference
if (CORBA::is_nil(mFile_var.in()))
{
    cout << "File.activateFile:invalid reference" << endl;
}
```

```

        message_var = CORBA::string_dup("invalid reference");
        ACE_TRY_THROW(CF::FileException(CF::CF_EBADF, message_var.in()));
    }

    // Give ownership of Servant to POA
    this->_remove_ref(ACE_TRY_ENV);
    ACE_TRY_CHECK;

```

6. The following code must be used to release the CORBA Object.

```

// release this Servant from the CORBA Environment
objectID_var = mPOA_var->servant_to_id(this, ACE_TRY_ENV);
ACE_TRY_CHECK;

mPOA_var->deactivate_object(objectID_var.in(), ACE_TRY_ENV);
ACE_TRY_CHECK;

```

When calling the POA.deactivate_object operation, several things occur. The POA does not allow for any additional CORBA calls to occur on the CORBA Object. The pending CORBA calls are processed. When there are no more CORBA calls to be processed, the CORBA Object is removed from the active object map and the servant destructor is called.

CORBA::Any

The CORBA::Any is a special data type unique to CORBA. In order to place data into a CORBA::Any type you must use the insertion operator (<<=), and to extract data from a CORBA::Any type you must use the extraction operator (>>=).

CORBA::Any types allocate/deallocate memory on their own. In this way, they act very much like _var types, and deallocate themselves when they go out of scope. Because a CORBA::Any type will try to deallocate itself, it is important not to extract data to a _var type which will try to deallocate the same memory space and will possibly cause a core dump. Always extract to _ptr types.

CORBA::Any types should be treated as read-only, as well as the _ptr type that it is extracted to. If a local copy is needed, or extracted data needs to be modified, use the _duplicate() function to copy the _ptr to a _var type and work with the _var type.

ACE/TAO

Exception Handling

ACE/TAO provides exception handling macros that accommodate exceptions whether the application environment does or not. These are being utilized by the OrcaCF to help ensure platform independence. Three of the most used Try blocks in the OrcaCF are listed below for reference.

Simple Try Block

```

ACE_TRY
{
    ...
    ACE_TRY_CHECK;

    ...
    ACE_TRY_CHECK;
}

```



```

        if (error)
        {
            ACE_TRY_THROW(CF::InvalidFileName(errno, errmsg));
        }

    } // end ACE_TRY

ACE_CATCH(CORBA::SystemException, exSystem)
{
    ACE_THROW(CORBA::SystemException);
}

ACE_CATCH(CF::FileException, exFile)
{
    ACE_RE_THROW;
}

ACE_ENDTRY;

```

Multiple Try Block

```

ACE_TRY_EX(block1)
{
    ...
    ACE_TRY_CHECK_EX(block1);

} // end ACE_TRY

ACE_CATCH(CORBA::SystemException, exSystem)
{
    ACE_THROW(CORBA::SystemException);
}

ACE_ENDTRY;

ACE_TRY_EX(block2)
{
    ...
    ACE_TRY_CHECK_EX(block2);
} // end ACE_TRY

ACE_CATCH(CORBA::SystemException, exSystem)
{
    ACE_THROW(CORBA::SystemException);
}

ACE_ENDTRY;

```

Try Block with Return Value

```

ACE_TRY
{
    ...
    ACE_TRY_CHECK;
}

```

```

...
ACE_TRY_CHECK;

if (error)
{
    ACE_TRY_THROW(CF::InvalidFileName(errno, errmsg));
}

} // end ACE_TRY

ACE_CATCH(CORBA::SystemException, exSystem)
{
    ACE_THROW_RETURN(CORBA::SystemException, my_var._retn());
}

ACE_ENDTRY;

ACE_CHECK_RETURN(my_var._retn());

```

Note: According to the TAO Developer's Guide, version 1.2a (p121):

“In environments that have C++ exceptions, ACE_CHECK and ACE_CHECK_RETURN are identical, as are ACE_THROW and ACE_THROW_RETURN. In environments that do not have native C++ exceptions, however, these macros give us the ability to write code that compiles successfully and functions properly.”

Basically, C++ exceptions must be disabled for the ACE_CHECK_RETURN and ACE_THROW_RETURN to function properly (i.e. return a value).

XML

Symbol Reference

Table 10-3 shows a list of XML symbols and associated definitions that are used within the DTDs provided with SCA 2.2. These definitions are provided for quick reference and ease of interpretation of the XML used with the OrcaCF. The table below represents only a small subset of the XML symbols and definitions.

Symbol	Definition
*	0..n relationship
+	1..n relationship
?	0..1 relationship
	Select one of several elements
EMPTY	Element can contain only attributes
ID	Represents the <code>ID</code> attribute type defined in the XML 1.0 Recommendation. The <code>ID</code> must be a no-colon-name (NCName) and must be unique within an XML document. This data type is derived from <code>NCName</code> .
PCDATA	PCDATA is text occurring in a context in which markup and entity references may occur. PCDATA describes the simplest XML structure of all, namely, plain text enclosed between a tag pair.
CDATA	An attribute of CDATA type can contain any character if it conforms to well formedness constraints. Everything inside a CDATA section is ignored by the parser. If your text contains a lot of "<" or "&" characters - as program code often does - the XML element can be defined as a CDATA section. A CDATA section starts with "<![CDATA[" and ends with "]]>". A CDATA section cannot contain the string "]]>", therefore, nested CDATA sections are not allowed. Also make sure there are no spaces or line breaks inside the "]]>" string.

Table 10-3 XML Symbols

Appendix C. Compilation Build Procedures

Scope

This document will provide general guidance for OrcaCF development using Red Hat Linux. However, it is not written for the novice user as it assumes that the user has a working knowledge of the Linux/Unix operating system and standard programming principles.

Document Overview

The remaining sections of this document are organized in the following manner:

Section 0 provides a list of references used to create this document.

Section 0 provides a list of hardware and software requirements for compiling and testing OrcaCF v1.1.0.

Section 0 provides detailed installation instructions for ACE/TAO and Xerces, as well as System Environment preparation.

Section 0 provides detailed installation instructions for OrcaCF v1.1.0.

Relationship to Other Documents

This document is an appendix to the OrcaCF v1.1.0 Software User's Manual (SUM), and provides a standard set of compilation, build and test procedures for the OrcaCF.

References

OrcaCF_QRG_v1_1_0.pdf: L-3 Communications GSI, June 2004.

Requirements

Hardware

The following is the system configuration of our test machine:

CPU - Intel(R) Pentium(R) 4 2.60GHz

RAM - 1024MB ECC DDR SDRAM

Display - NVIDIA Quadro 4 (generic)

Sound - SoundBlaster Audigy 2

Storage - 80GB IDE hard drive with ext3 filesystem

Network Interface - Intel 82540EM Gigabit Ethernet Controller

Software

The following software packages, along with their version numbers, were installed on our test machine:

Desktop (OS) – Red Hat Linux 9.0

- Linux Kernel 2.4.20-8
- KDE 3.1-10

Net/File Browser – Konqueror 3.1-12

IDE – KDevelop 2.1.5

Compiler – gcc 3.2.2

make 3.79.1

XML Parser – Apache Xerces C++ 2.5.0

CORBA ORB - ACE v5.4/ TAO v1.4

**NOTE: It is up to the developer to ensure they have a properly configured sound card and/or onboard sound, in order to run the Sound Demo we include with the OrcaCF. We have successfully tested a number of different sound cards and onboard sound. The Sound Demo was implemented using the free version of the Open Sound System (OSS/Free), which was included in the Red Hat distributions 9.0 and earlier. The Red Hat sponsored Fedora distributions have begun removing the OSS in favor of the Advanced Linux Sound Architecture (ALSA). Fedora Core 1 uses OSS as the default sound interface. Fedora Core 2 switched to ALSA. The Sound Demo DOES NOT work with ALSA.*

Dependencies and Preparation

ACE/TAO and XERCES are dependencies for OrcaCF v1.1.0. The following sections include the installation instructions for ACE/TAO and XERCES, as well as the System Environment preparation for building the OrcaCF v1.1.0. The recommended installation paths are provided in the instructions and should be followed, as the installation locations are key for the building and testing of the OrcaCF projects. A couple of notes about the instructions in this document; we use <ENTER> as an indication for the user to press the enter key on their keyboard, and we denote directory name differences by using < > characters (eg. <username> refers to the user's Linux login name).

Install ACE/TAO

The following are the procedures for installing ACE/TAO on a Linux machine. If ACE/TAO is currently installed on your machine, you may skip to section 0 below.

Download a stable release of ACE/TAO:

The version we currently use is ACE 5.4 with TAO 1.4. The file *ACE-5.4+TAO-1.4.tar.gz* is available for download via the following link http://deuce.doc.wustl.edu/old_distribution/. Save this file into a directory of your choosing.

Place a copy of the file in installation root directory:

Move or make a copy of the downloaded file and place it in the installation root directory of your choosing. We currently have “/usr/local/TAO” as our installation root directory.

Note: Administrative privileges may be required in order to perform the above copy operation, depending on the configuration of the Linux OS. We chose to download and install ACE/TAO while logged in as “root”.

Decompress the Zipped File:

Extract the ACE/TAO package by right-clicking on the file and choosing *Extract Here...*, or ..., you may open a console (terminal) window and change to your chosen installation root directory, in our case “/usr/local/TAO”. The following console commands will decompress the package and create a new subdirectory within the installation root directory labeled “ACE_wrappers”.

```
gunzip ACE-5.4+TAO-1.4.tar.gz <ENTER>
tar -xvf ACE-5.4+TAO-1.4.tar <ENTER>
```

Note: Using the “ls” command will verify that the new subdirectory “ACE_wrappers” was created during the unzipping process.

Setup ACE/TAO Environment Variables:

The BASH shell is our shell of choice for all installation and testing procedures. The setting up of BASH environment variables is done by modifying the .bashrc file.

Locate your .bashrc file using the Konqueror browser. This file is generally located in the user’s home directory, “/home/<username>” where “<username>” indicates the user’s login name. If you do not see this file in your home directory you may need to enable the visibility of hidden files. To do so, within the Konqueror browser, select *Show Hidden Files* from the View menu.

Open your .bashrc file using your text editor of choice.

Add the following lines at the end of your .bashrc file if they don’t already exist:

```
export ACE_ROOT=/usr/local/TAO/ACE_wrappers
export TAO_ROOT=$ACE_ROOT/TAO
export PATH=$PATH:$TAO_ROOT/TAO_IDL
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$ACE_ROOT/ace
```

Note: The paths listed in the above statements represent the paths chosen for our installation. Make the necessary changes needed for your installation path.

Below is a picture of a valid .bashrc file consisting of the ACE/TAO environment variables representative of our installation:

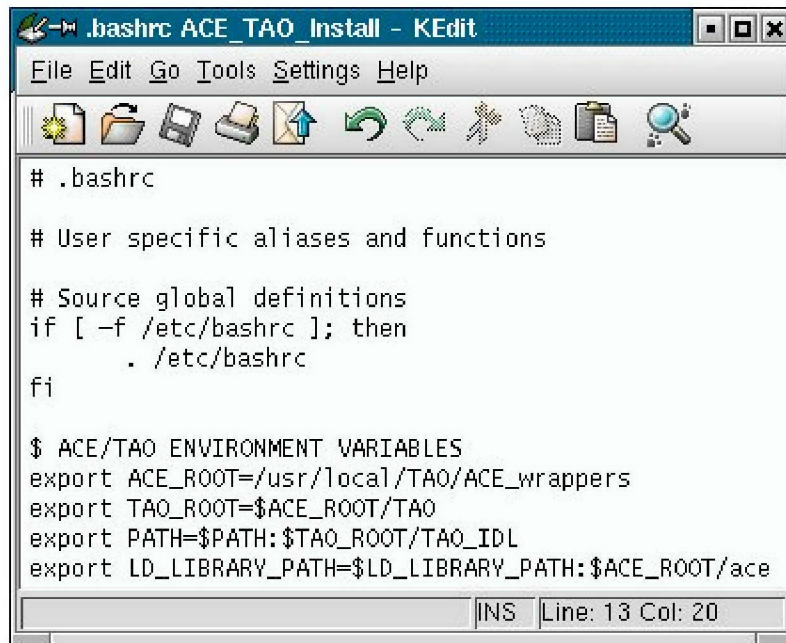


Figure 10-1 bashrc File with ACE/TAO Environment Variables

Save the file.

Log out and then log back in for the above changes to take effect.

Setting Configuration Files

Next, two header files pertaining to the platform and compiler used must be created.

1. Using a console window, change to the “ACE_ROOT/ace” directory. The “ACE_ROOT” variable was defined when setting up the ACE/TAO environment variables. The following command quickly takes you to the desired directory.

```
cd $ACE_ROOT/ace <ENTER>
```

2. Next, create a symbolic link to the proper config file by typing the following command:

```
ln -s config-linux.h config.h <ENTER>
```

3. Change to “ACE_ROOT/include/makeinclude” directory.

```
cd $ACE_ROOT/include/makeinclude <ENTER>
```

4. Finally, create the last required link by typing the following command:

```
ln -s platform_linux.GNU platform_macros.GNU <ENTER>
```

Building the Sources

Now that the environment is set and the appropriate header files are defined, it's time to build the ACE/TAO source files. We chose to build everything that came with the ACE/TAO download, including the large number of examples. At each step listed below, it may take an extended period of time for the sources to build, depending on the speed of the machine you are using. The entire build process took over

three hours to complete on our test machine. Experienced ACE/TAO users may go in and build only the binaries they need.

1. Using a console window, change to the “ACE_ROOT/ace” directory.
`cd $ACE_ROOT/ace <ENTER>`
2. Type the following command:
`make <ENTER>`
3. Once the ACE build is complete, change to the “ACE_ROOT/apps/gperf” directory.
`cd $ACE_ROOT/apps/gperf <ENTER>`
4. Type the following command:
`make <ENTER>`
5. Once the gperf build is complete, change to the “TAO_ROOT” directory.
`cd $TAO_ROOT <ENTER>`
6. Type the following command:
`make <ENTER>`

****WARNING:** We encountered compile/link errors during the build process of ACE v5.4/TAO v1.4. The makefiles do not appear to build the projects in the proper order, therefore, we get link errors because certain libraries are not built yet. There is also a subdirectory listed in one of the makefiles that does not exist. Instructions for fixing these problems are listed below:*

- a. After you have started the last build step (step 6 above), you may encounter an error searching for the Kokyu (-lKokyu) shared library. Change to the Kokyu directory and run make:

```
cd $ACE_ROOT/Kokyu <ENTER>
make <ENTER>
```

- b. After the Kokyu library finishes building, go back to the “TAO_ROOT” directory and re-run make:

```
cd $TAO_ROOT <ENTER>
make <ENTER>
```

- c. Another error you may encounter is a missing ACEXML_Parser (-lACEXML_Parser) shared library. Change to the ACEXML directory and run make:

```
cd $ACE_ROOT/ACEXML <ENTER>
make <ENTER>
```

- d. After the ACEXML_Parser library finishes building, go back to the “TAO_ROOT” directory and re-run make:


```
cd $TAO_ROOT <ENTER>
make <ENTER>
```

- e. The last error we came across was a directory listing in one of the makefiles that does not exist. The Makefile is located in TAO_ROOT. The error is shown in the Makefile below:

```
#-----
#          Local macros
#-----
INFO      = README \
           VERSION

DIRS      = tao \
           TAO_IDL \
           tests \
           orbsvcs \
           examples \
           performance-tests \
           utils \
           docs/tutorials/Quoter \
           CIAO
```

The directory CIAO shown above does not exist and needs to be deleted from the \$TAO_ROOT/Makefile. Using your preferred editor delete the CIAO directory listing from the Makefile and save it. (Don't forget to delete the "\" mark from the Quoter directory listed above CIAO)

- f. After fixing and saving the Makefile, go back to the "TAO_ROOT" directory and re-run make:

```
cd $TAO_ROOT <ENTER>
make <ENTER>
```

If you encounter any other compile/build errors during ACE/TAO installation, please refer to the ACE/TAO website for assistance (<http://www.cs.wustl.edu/~schmidt/TAO.html>). For other users wishing to develop CORBA applications (eg. OrcaCF/SCA) using ACE/TAO, they must setup their respective ACE/TAO environment variables, as described in Section 0.

Install Xerces

The following are the procedures for installing Xerces C++ on a Linux machine. If Xerces is currently installed on your machine, you may skip to Section 0.

Download a Stable Release of Xerces:

The version we currently use is Xerces C++ 2.5.0. The file `xerces-c-current.tar.gz` is available for download via the following link: <http://xml.apache.org/xerces-c/>. Save this file into a directory of your choosing.

Place a copy of the file in installation root directory:

Move or make a copy of the downloaded file and place it in the installation root directory of your choosing. We placed the downloaded file in `/usr/local` (installation root) with the intent of having our final Xerces installation directory be `/usr/local/xerces-c-src_2_5_0`. The `"xerces-c-src_2_5_0"` subdirectory is created during the unzipping process that follows.

Note: Administrative privileges may be required in order to perform the above copy operation, depending on the configuration of the Linux OS. We chose to download and install Xerces while logged in as root.

Decompress the Zipped file:

Extract the `xerces-c-current.tar.gz` package by right-clicking on the file and choosing *Extract Here...*, or..., you may open a console (terminal) window and change to your chosen installation root directory, in our case `/usr/local`. The following console commands will decompress the package and create a new subdirectory within the installation root directory labeled `"xerces-c-src_2_5_0"`. Proceed with the following commands:

```
gunzip xerces-c-current.tar.gz <ENTER>
tar -xvf xerces-c-current.tar <ENTER>
```

Note: Using the "ls" command will verify that the new subdirectory "xerces-c-src_2_5_0" was created during the unzipping process.

Setup Xerces Environment Variables:

The BASH shell is our shell of choice for all installation and testing procedures. The setting up of BASH environment variables is done by modifying the `.bashrc` file.

Locate your `.bashrc` file using the Konqueror browser. This file is generally located in the user's home directory, `/home/<username>`. If you do not see this file in your home directory you may need to enable the visibility of hidden files. To do so, within the Konqueror browser, select *Show Hidden Files* from the View menu.

Open your `.bashrc` file using your text editor of choice.

Add the following lines at the end of your `.bashrc` file if they don't already exist:

```
export XERCESCROOT=/usr/local/xerces-c-src_2_5_0
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$XERCESCROOT/lib
```

Note: The paths listed in the above statements represent the paths chosen for our installation. Make the necessary changes needed for your installation path.

Below is a picture of a valid `.bashrc` file consisting of the ACE/TAO and Xerces environment variables representative of our installation:

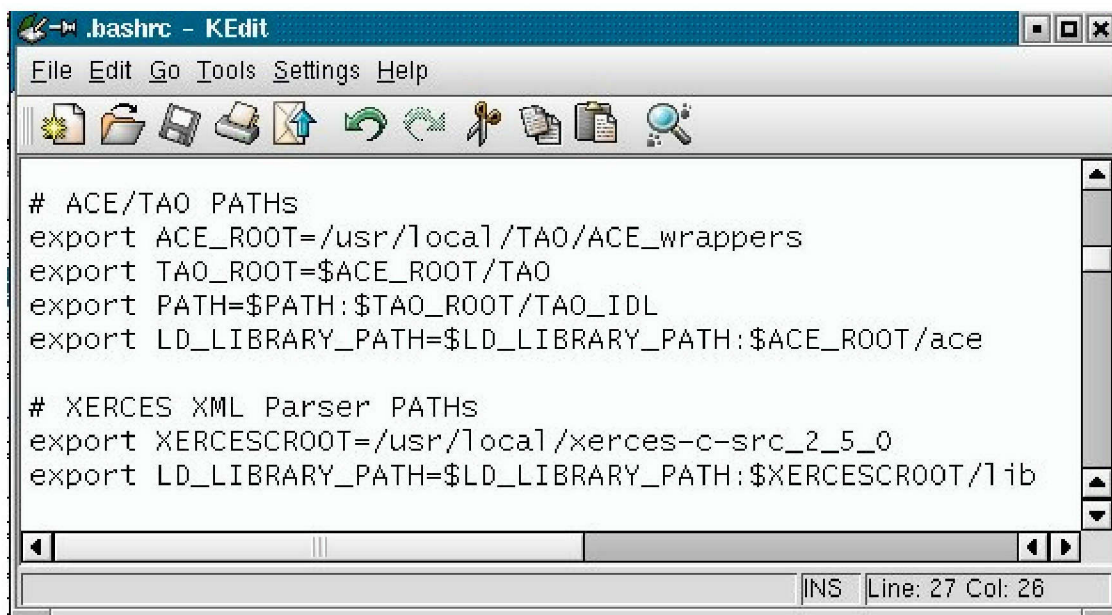


Figure 10-2 bashrc File with ACE/TAO and Xerces Environment Variables

Save the file.

Log out and then log back in for the above changes to take effect.

Building the Sources:

Now that the environment is set, it's time to build the Xerces source files.

The Xerces website has detailed and advanced build instructions for more advanced users. We will list our simple build instructions below:

1. Using a console window, change to the "XERCESSROOT/src/xercesc" directory.
`cd $XERCESSROOT/src/xercesc <ENTER>`
2. Type the following command:
`./runConfigure -plinux <ENTER>`
3. Type the following command:
`make <ENTER>`

If you encounter any compile/build errors during Xerces installation, please refer to the Xerces website (<http://xml.apache.org/xerces-c/index.html>).

OrcaCF installation

The following are the procedures for setting up a Linux machine for building the OrcaCF source code.

Place OrcaCF Package in installation root directory

The name of the file containing the source code is `OrcaCF_v1_1_0_source.tar.gz`. Place a copy of the file in the installation root directory of your choosing. When testing our material, we choose the

installation root directory to be our “HOME” directory, or explicitly “/home/<username>”, with the intent of having our final OrcaCF installation directory be “/home/<username>/OrcaCF”. The “OrcaCF” subdirectory is created during the unzipping process that follows.

Decompress the Zipped File

Extract the OrcaCF_v1_1_0_source.tar.gz package by right-clicking on the file and choosing *Extract Here...*, or ..., you may open a console (terminal) window and change to your chosen installation root directory, in our case “/home/<username>”. The following console commands decompress the package and create a new subdirectory within the installation root directory labeled “OrcaCF”. Proceed with the following commands:

```
gunzip OrcaCF_v1_1_0_source.tar.gz <ENTER>
tar -xvf OrcaCF_v1_1_0_source.tar <ENTER>
```

Note: Using the “ls” command will verify that the new subdirectory “OrcaCF” was created during the unzipping process.

Environment Variables

The BASH shell is our shell of choice for all installation and testing procedures. The setting up of BASH environment variables is done by modifying the .bashrc file.

Locate your .bashrc file using the Konqueror browser. This file is generally located in the user’s home directory, “/home/<username>” where “<username>” indicates the user’s login name. If you do not see this file in your home directory you may need to enable the visibility of hidden files. To do so, within the Konqueror browser, select *Show Hidden Files* from the View menu.

Open your .bashrc file using your text editor of choice.

Add the following lines at the end of your .bashrc file if they don’t already exist:

```
export ORCACF_ROOT=$HOME/OrcaCF
export PATH=$PATH:$ORCACF_ROOT/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$ORCACF_ROOT/lib
export NS_OPTIONS="--ORBdottedDecimalAddresses 1 -ORBEndpoint \
iiop://<hostname>:<port> -m 0 -d"
export CF_OPTIONS="--ORBdottedDecimalAddresses 1 -ORBInitRef \
NameService=corbaloc::<hostname>:<port>/NameService"
```

Note: The paths listed in the above statements represent the paths chosen for our installation. Make the necessary changes needed for your installation path.

The NS_OPTIONS and CF_OPTIONS environment variables contain parameters that are unique and must be set by the user. The <hostname> is the hostname of the machine that OrcaCF is executed on and the <port> is a unique port number that will be used by OrcaCF executables. The NS_OPTIONS contains the ORBEndpoint parameter that tells the ORB to listen for requests on the interface specified by the endpoint. Endpoints are specified using a URL style format. An example of an IIOP endpoint is:

```
iiop://localhost:9999
```

The standard installation of Linux distributions installs a network LOOPBACK interface called “localhost” with an IP address of 127.0.0.1. Using “localhost” is recommended for anyone not

familiar with networking. If you have altered the “localhost” interface in any way, the application may not work properly. The `CF_OPTIONS` environment variable contains the `ORBInitRef` parameter, which is the ORB initial reference argument. This argument allows specification of an arbitrary object reference for an initial service which, in this case, is the Naming Service. The format is:

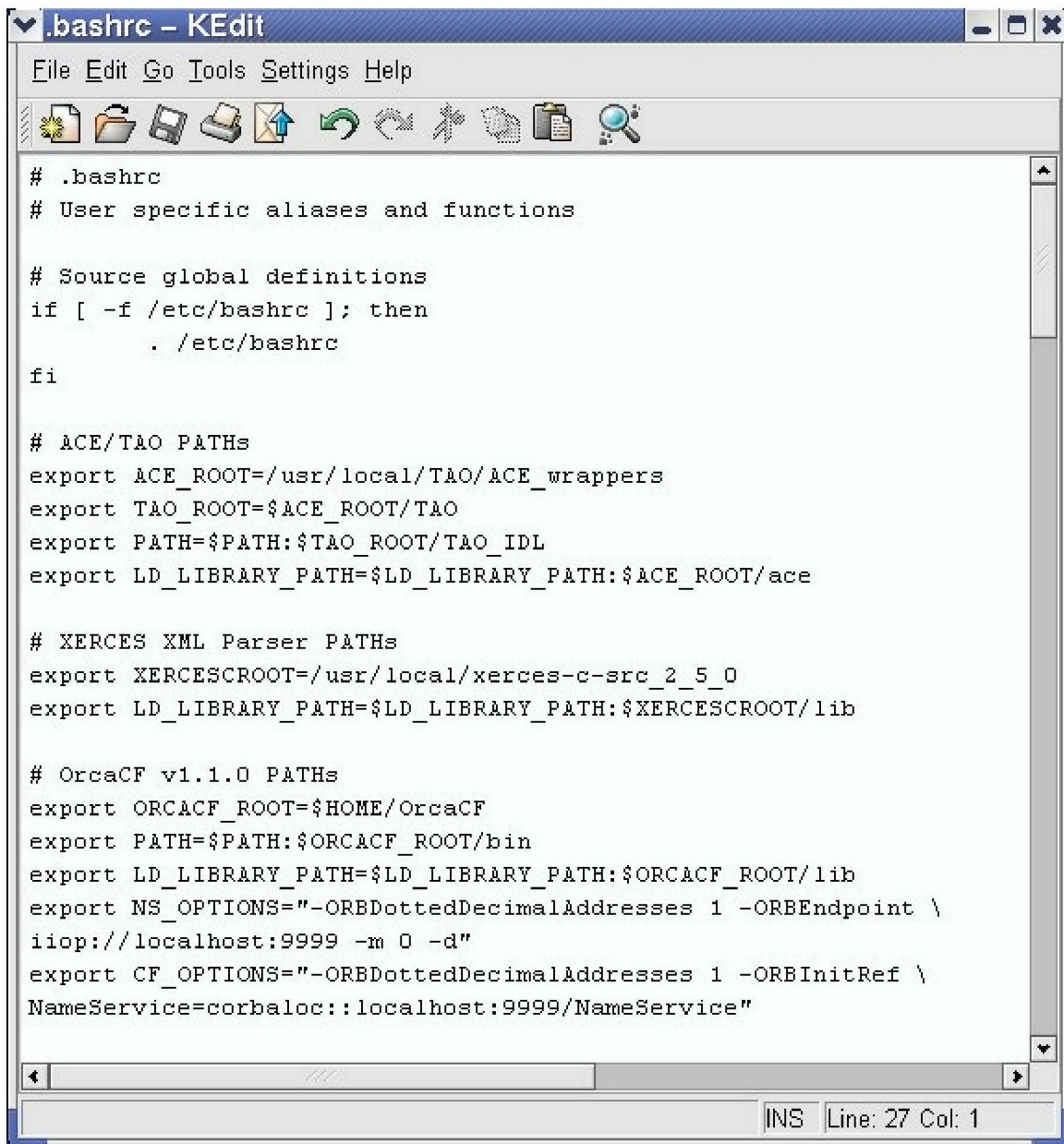
```
-ORBInitRef [ObjectID]=[ObjectURL]
```

Using “localhost” (recommended), the line would look like this:

```
-ORBInitRef NameService=corbaloc::localhost:9999/NameService
```

The “localhost” and “port” must match for proper CORBA communication.

Below is a picture of a valid `.bashrc` file consisting of the ACE/TAO, Xerces, and OrcaCF environment variables representative of our installation. For beginners, it is recommended that you use the PATHs shown below:



```
# .bashrc
# User specific aliases and functions

# Source global definitions
if [ -f /etc/bashrc ]; then
    . /etc/bashrc
fi

# ACE/TAO PATHs
export ACE_ROOT=/usr/local/TAO/ACE_wrappers
export TAO_ROOT=$ACE_ROOT/TAO
export PATH=$PATH:$TAO_ROOT/TAO_IDL
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$ACE_ROOT/ace

# XERCES XML Parser PATHs
export XERCECROOT=/usr/local/xerces-c-src_2_5_0
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$XERCECROOT/lib

# OrcaCF v1.1.0 PATHs
export ORCACF_ROOT=$HOME/OrcaCF
export PATH=$PATH:$ORCACF_ROOT/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$ORCACF_ROOT/lib
export NS_OPTIONS="-ORBdottedDecimalAddresses 1 -ORBEndpoint \
iiop://localhost:9999 -m 0 -d"
export CF_OPTIONS="-ORBdottedDecimalAddresses 1 -ORBInitRef \
NameService=corbaloc::localhost:9999/NameService"
```

Figure 10-3 Final bashrc File

Save the file.

Log out and then log back in for the above changes to take effect.

Building the Sources:

Now that the environment is set, it's time to build the OrcaCF source files.

1. Go to the `ORCACF_ROOT/src` directory and make sure the `OrcaCF_install` script has execute permissions.

Type the following commands:

```
cd $ORCACF_ROOT/src <ENTER>
```

```
ll <ENTER>
```

The install script should now be listed with its permissions. Execute permissions should be listed (-x) in at least one spot to the far left. It should look similar to the following:

```
-rwxr-xr-x 1 <username> users 1756 <date><time> OrcaCF_install
```

If you do not have execute permissions, or do not understand how to tell, do a web search on the `chmod` command in Linux, or simply type:

```
man chmod <ENTER>
```

in the console window. This will show you how to change permissions on a file. If the permissions look ok, move on to the next step.

2. Run the `OrcaCF_install` script.

Type the following command:

```
./OrcaCF_install <ENTER>
```

This will build all the required components of the OrcaCF. Once the build process is complete the OrcaCF is installed and ready to be run. Refer to the `OrcaCF_QRG_v1_1_0.doc` (Quick Reference Guide) for instructions on how to run the OrcaCF Sound Demo.

Summary

This document is a guide for building the OrcaCF shared libraries and executables, and for setting up the OrcaCF development structure. If you have any questions, comments, or feedback, please email to OrcaCF.gsi@L-3com.com.

Appendix D. Software Design

Scope

This document is the software design description for the Open Radio Communication Architecture Core Framework (OrcaCF) v1.1.0.

Overview

This Appendix to the Software Users Manual (SUM) provides the “as-built” design of the OrcaCF v1.1.0. This document includes the design diagrams developed in the Unified Modeling Language (UML). The tool used to develop the UML diagrams is Rational Rose, Rose Enterprise Edition, Release Version 2002-05-20. The remaining sections of this appendix are organized in the following manner.

Section 0 is an overview of the OrcaCF development, providing information about the operating environment around which the OrcaCF has been designed.

Section 0 provides an overview of the OrcaCF project, defines the OrcaCF packages, and shows the relationship between the OrcaCF packages.

OrcaCF Operating Environment

The OrcaCF has been designed around the Open Source model, utilizing open source commercial off the shelf components. Figure 10-4 below shows the notional relationship between the SCA Operating Environment components selected or built for the OrcaCF.

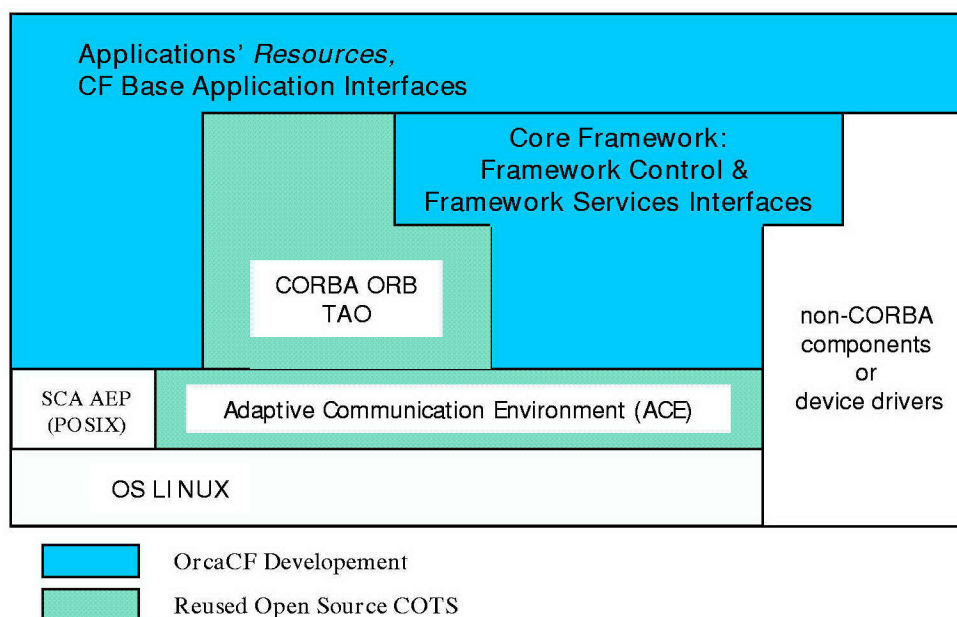


Figure 10-4: OrcaCF Design of SCA Operating Environment

Operating System

The OrcaCF has been designed and built on the Linux operating system. Linux is a POSIX .1 compliant operating system, and it also satisfies the SCA AEP requirements.

The current version of the OrcaCF has been designed and built on Red Hat 9.0 utilizing the Linux Kernel version 2.4.20-8.

Middleware (CORBA)

The OrcaCF utilizes The ACE Orb (TAO) to satisfy the middleware and CORBA requirements of the SCA. TAO is built on top of the Adaptive Communication Environment (ACE).

ACE provides a set of C++ wrappers and framework components that perform common communication software tasks across multiple OS platforms. This enables TAO to be used on multiple OSs (e.g. Windows, POSIX, VxWorks, etc.). The OrcaCF also reuses ACE within its packages to take advantage of the portability. On a POSIX OS, ACE uses the POSIX.1 calls that are defined in the SCA AEP.

Both ACE and TAO are freely available open-source products, which fit the overall design of the OrcaCF. The current version of the OrcaCF has been built around ACE version 5.4 and TAO version 1.4.

Naming Service

TAO's CosNaming ORB service satisfies the requirements set forth within the SCA. This component is reused by the OrcaCF to meet the SCA requirements for a Naming Service. TAO's CosNaming component is a shared library. TAO also provides an executable called Naming_Service used for launching the CosNaming ORB service. This executable is also reused by the OrcaCF.

Event Service

TAO's CosEvent ORB service satisfies the requirements set forth within the SCA. This component is reused by the OrcaCF to meet the SCA requirements for an Event Service. TAO's CosEvent component is a shared library.

The SCA also provides a set of standard event interfaces defined in IDL. This IDL has been compiled by the TAO IDL compiler into C++ source code and is located within the OrcaCF SCA Interface package. The OrcaCF SCA Interface package is described in greater detail below.

The combination of TAO's CosEvent component and the OrcaCF SCA Interface component are utilized to satisfy the Event Service requirements specified within the SCA.

Log Service

The SCA defines a Log Service to be provided with the Operating Environment. While this Log Service is optional, the OrcaCF provides a C++ Log Service component designed and built around TAO, ACE and Linux.

The OrcaCF Log Service is an executable component implemented within the OrcaCF Log Service package. The OrcaCF Log Service package is described in greater detail below.

The SCA also provides a standard set of APIs for a Log Service defined in IDL. This IDL has been compiled by the TAO IDL compiler into C++ source code and is located within the OrcaCF SCA Interface package. The OrcaCF SCA Interface package is described in greater detail below.

The combination of the OrcaCF Log Service component and the OrcaCF SCA Interface component are utilized to satisfy the Log Service requirements specified within the SCA.

Core Framework

The SCA defines the Core Framework's interfaces and behavior. These interfaces are utilized for implementing the components of an SCA compliant software radio. The SCA provides the IDL for these defined interfaces. The relationships between the SCA-defined class interfaces are shown in Figure 10-5 below.

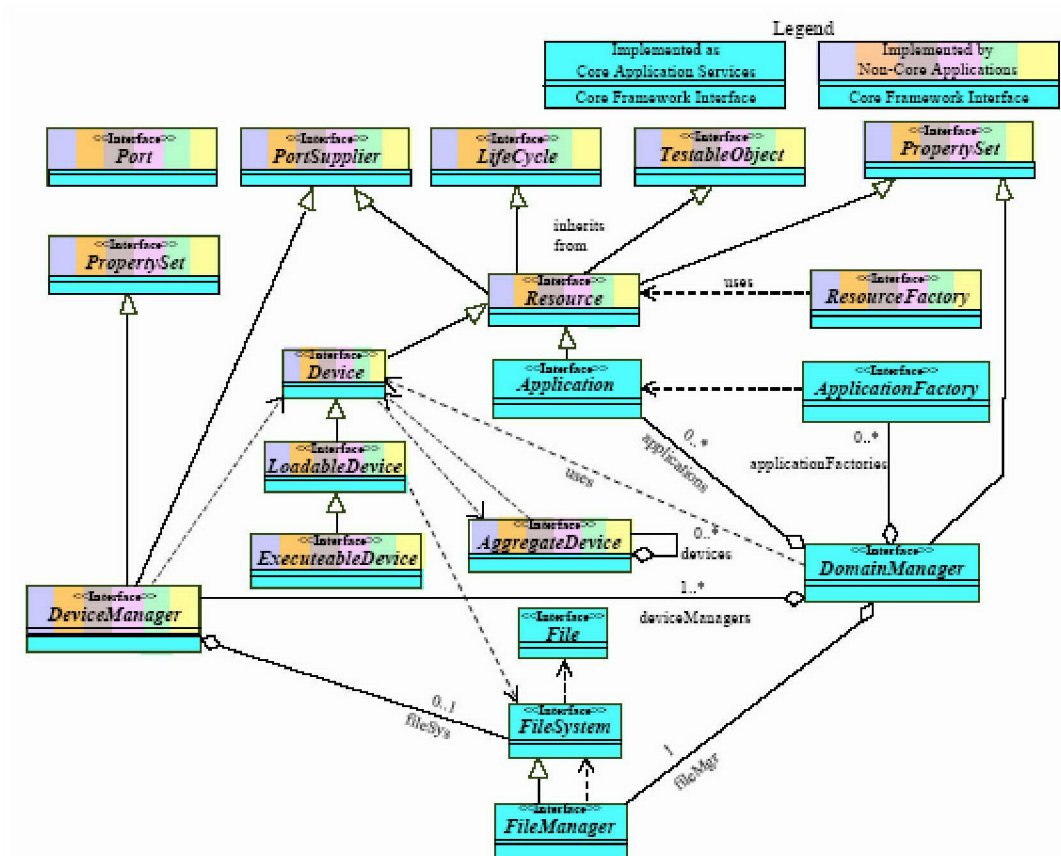


Figure 10-5: Core Framework Class Relationships defined in IDL

The Core Framework IDL has been compiled by the TAO IDL compiler into C++ source code and is located within the OrcaCF SCA Interface package. The OrcaCF SCA Interface package is described in greater detail below.

The implementations of the SCA components defined by the SCA have been implemented within the OrcaCF Project and are described in the sections that follow

OrcaCF Project

The OrcaCF project is designed in a modular fashion consisting of several components or subprojects. These components consist of executables, shared libraries, and external third-party utilities. Each of the components is described in greater detail in the sections that follow.

The design of the OrcaCF project has been diagrammed using Rational Rose. The Rational Rose project for the OrcaCF has been set up so that each component within the model corresponds to a different subproject or third-party component. Figure 10-6 shows the component view for the Rational Rose project.

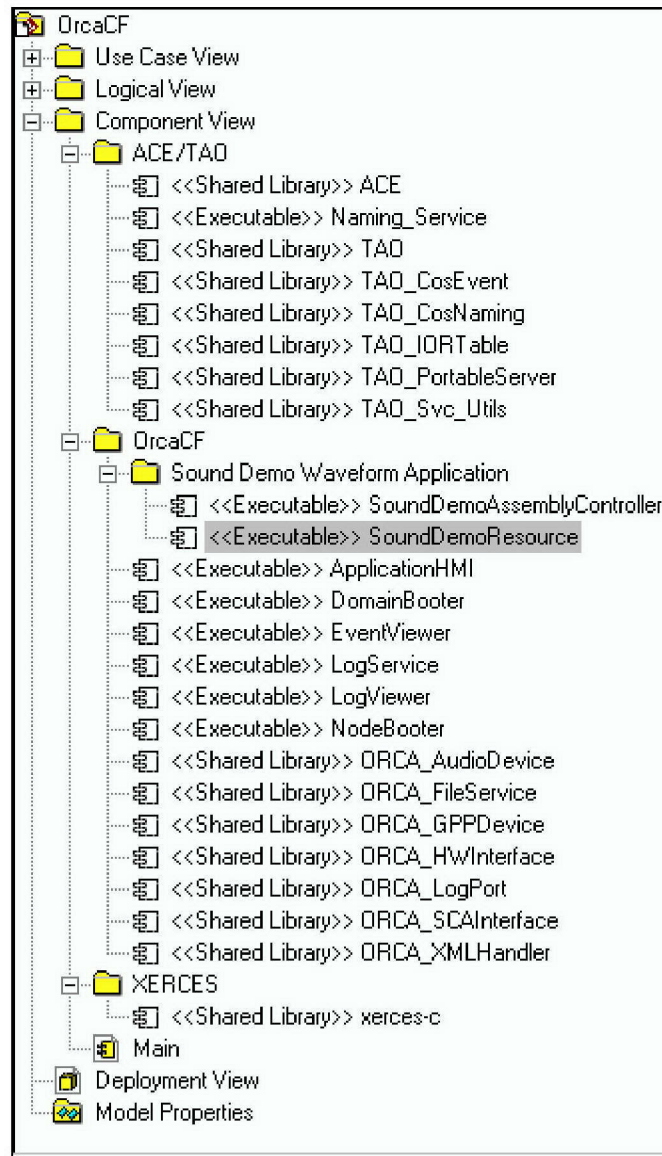


Figure 10-6 OrcaCF Rational Rose Project

The OrcaCF package shown in figure 3-1 shows the components that have been developed for the OrcaCF project. This figure also indicates the stereotype of each component, which matches the <softpkg> <implementation> <code> <type> attribute defined in Appendix D of the SCA. The components shown in the ACE/TAO and XERCES packages are third-party components used by the OrcaCF.

Domain Booter

The Domain Booter is an executable component implementing the Core Framework interfaces for the Domain Manager, Application Factory and Application servant objects. The class diagrams for these servant objects can be seen in Figure 10-7 and Figure 10-8.

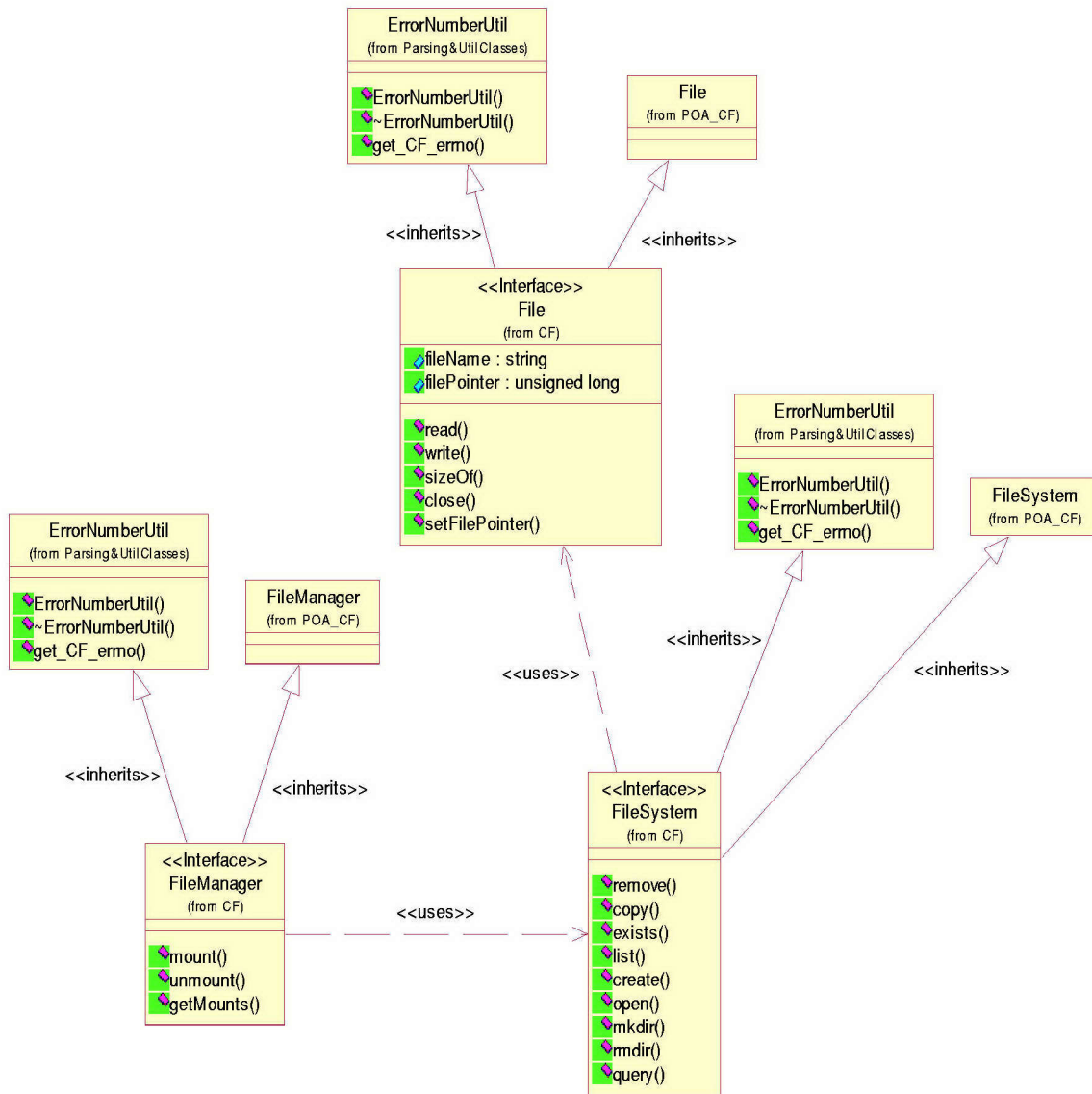


Figure 10-9: Class Diagram for File Service

The File Service component has been designed as a shared library so that it can be dynamically loaded and shared between the other components of the OrcaCF.

The File Service component also has internal and external dependencies to ACE, TAO, and SCA Interface.

SCA Interface

The SCA Interface is a shared library component providing the client stubs and server skeletons for the interfaces defined in Appendix C of the SCA. These interfaces were defined in IDL and contained in the files CF.idl, PortType.idl, Log.idl and StandardEvent.idl. All of the C++ code contained within this component has been auto-generated by running these IDL files through the TAO IDL compiler. Figure 10-10, Figure 10-11, Figure 10-12, and Figure 10-13 show the interfaces contained within each IDL file.

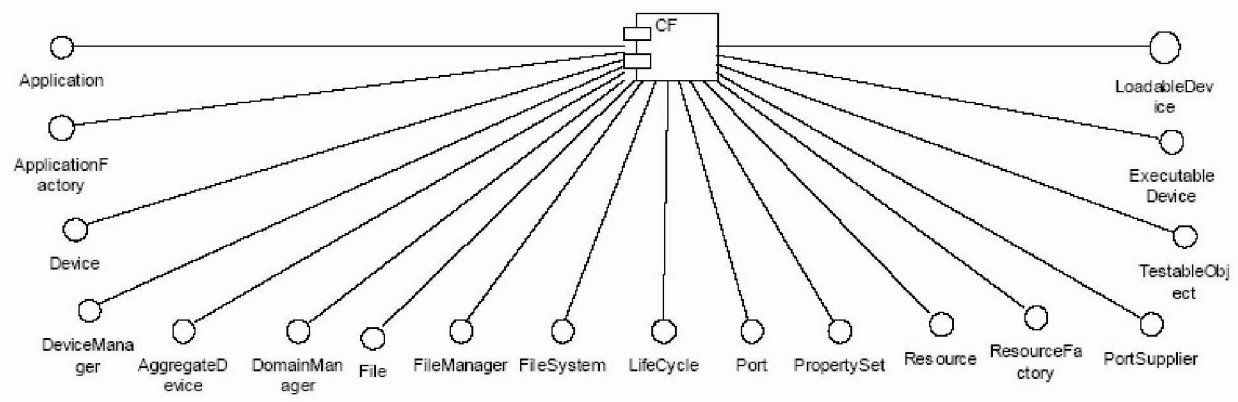


Figure 10-10: Interfaces Defined In CF.idl

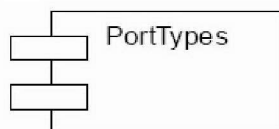


Figure 10-11: PortTypes.idl

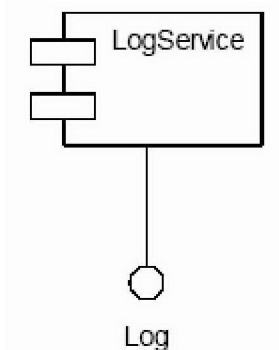


Figure 10-12: Interface Defined In Log.idl

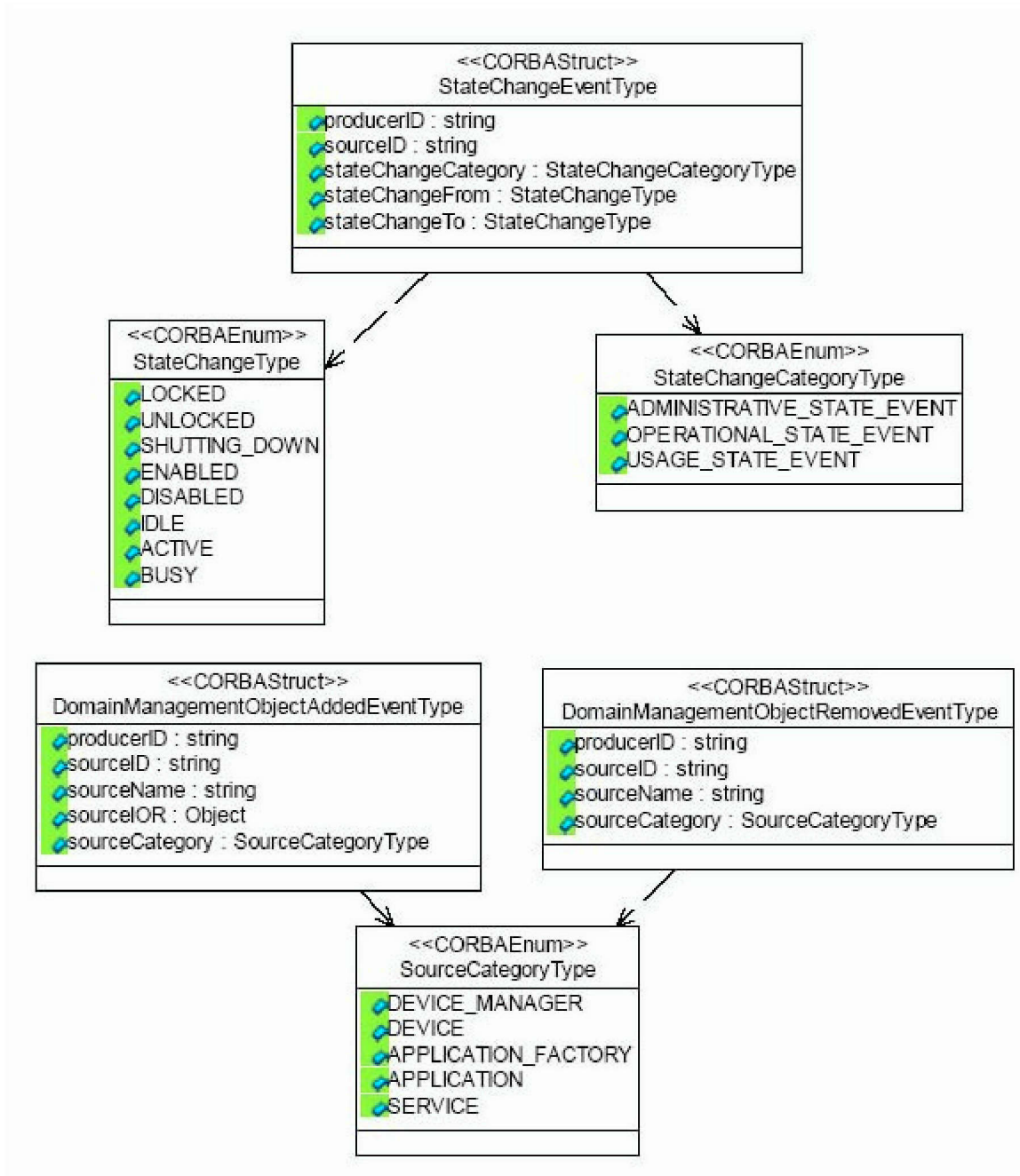


Figure 10-13: Interfaces Defined In StandardEvent.idl

The SCA Interface component has been designed as a shared library so that it can be dynamically loaded and shared between the other components of the OrcaCF. Almost all components of the OrcaCF are dependent on this shared library.

The SCA Interface has external dependencies to ACE and TAO.

Log Port

The Log Port is a shared library component implementing the Core Framework interface for the Port servant object. This object is a Core Framework uses port utilized by log producers within the OrcaCF to connect to and communicate with an SCA Log Service. The class diagram for this servant object can be seen in Figure 10-14.

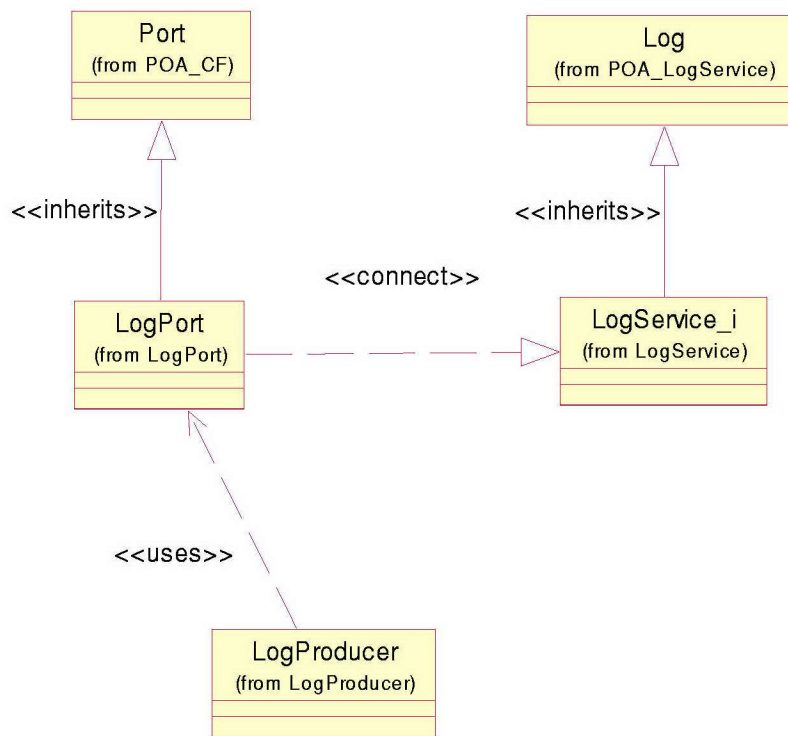


Figure 10-14 Log Port Class Diagram

This component has been designed as a shared library so that it can be dynamically loaded and utilized by all OrcaCF log producer objects.

The Log Port has internal and external dependencies to ACE, TAO and SCA Interface.

XML HANDLER

The XML Handler is a shared library component providing a common interface for accessing data contained within the XML defined by the SCA Domain Profile. An example of how the XML Handler is used by an OracCF component can be seen in Figure 10-15.

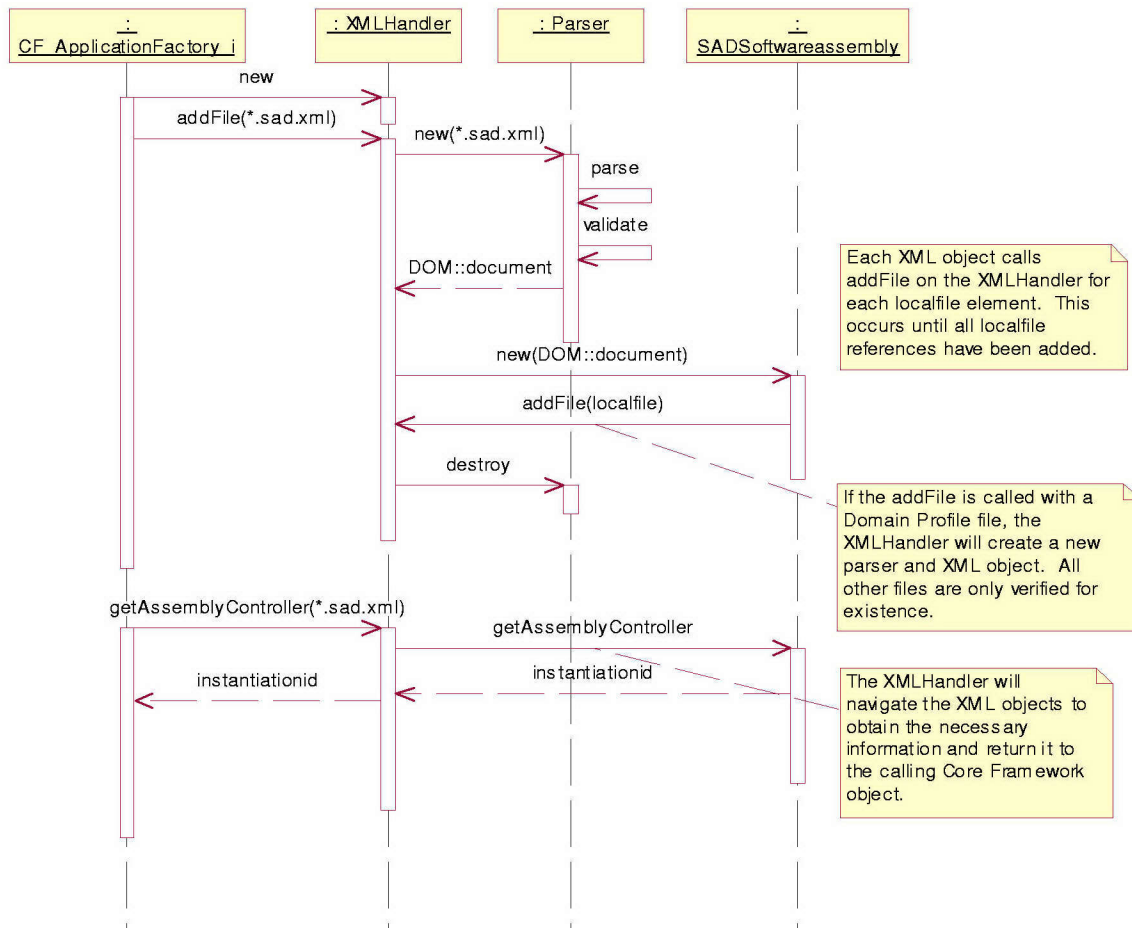


Figure 10-15: Example of XML Handler

The XML Handler component has been designed with a layered approach. The first layer is the XML business objects, which are auto-generated from the DTDs provided by the SCA Domain Profile. These objects store and provide access to the data extracted from the XML files. There are a set of classes associated with each Domain Profile DTD and are explained in greater detail in the sections that follow.

The Parser layer does all of the parsing and validation of the XML files. These files are validated against the DTDs provided with the OrcaCF. The Parser utilizes the XERCES parser to parse and validate the files. The XML files are parsed into a DOM Document. Once a file has been parsed and validated the XML Handler object uses the DOM Document to create the XML business objects.

All interaction between the Core Framework objects and the XML takes place through the XML Handler object. The XML Handler object uses the Parser object to parse and validate XML, and uses the XML business objects to store and navigate through the XML data. The XML Handler is responsible for extracting and formatting XML data for use by the Core Framework objects.

The Core Framework objects that use the XML Handler consist of the Domain Manager, Application Factory, Application and Device Manager. The class diagram for the XML Handler object can be seen in Figure 10-16.

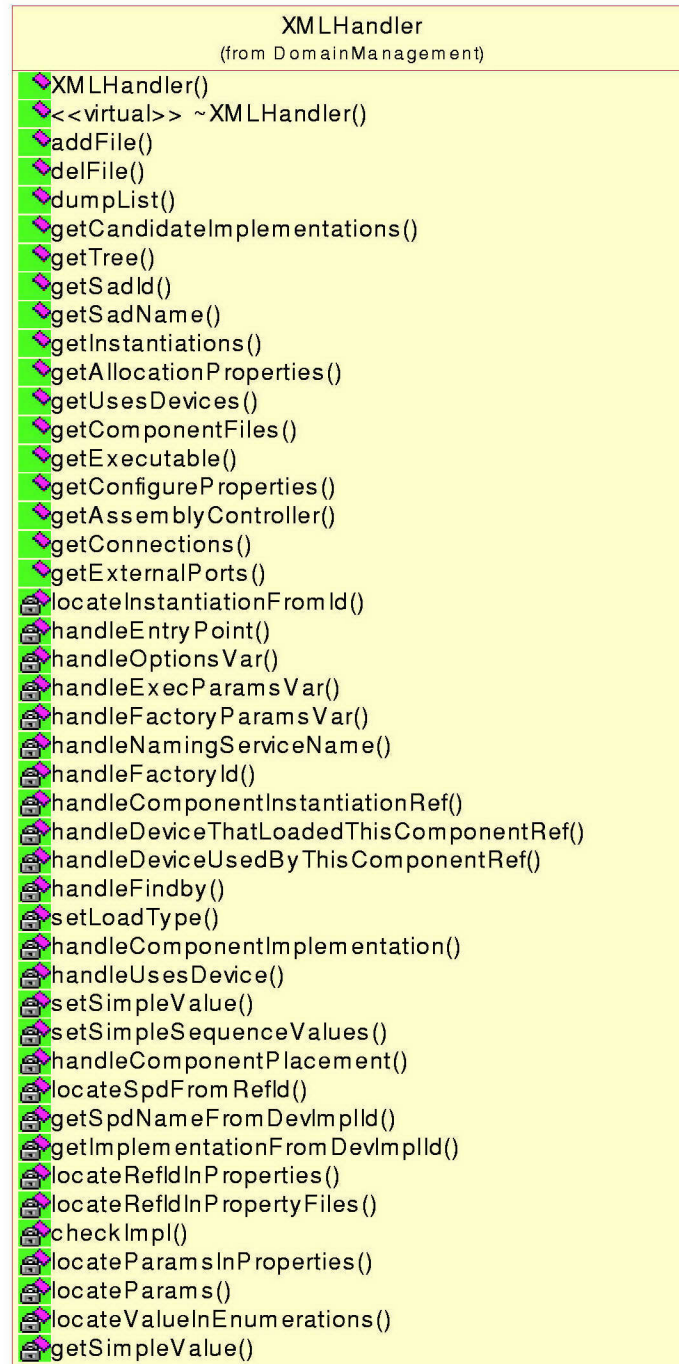


Figure 10-16: XML Handler Class

DCD XML Business Objects

The DCD XML business objects are derived from the SCA Device Configuration Descriptor (DCD) DTD. These classes are used to store and provide access to XML data within a DCD XML file. Each class represents a corresponding element definition inside the DCD DTD.

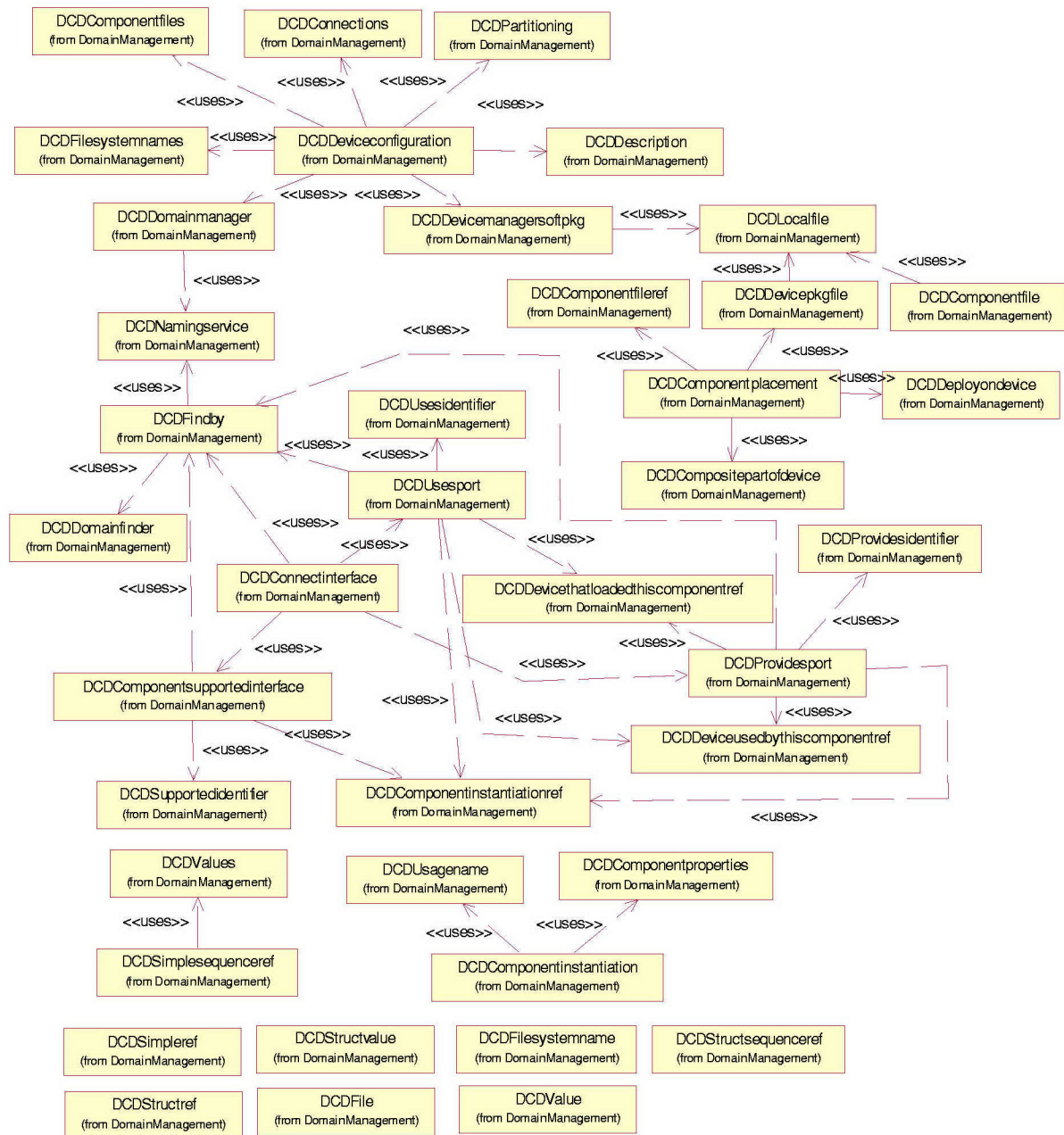


Figure 10-17: Class Diagram of DCD XML Business Objects

DMD XML Business Objects

The DMD XML business objects are derived from the SCA Domain Manager Configuration Descriptor (DMD) DTD. These classes are used to store and provide access to XML data with a

DMD XML file. Each class represents a corresponding element definition inside the DMD DTD.

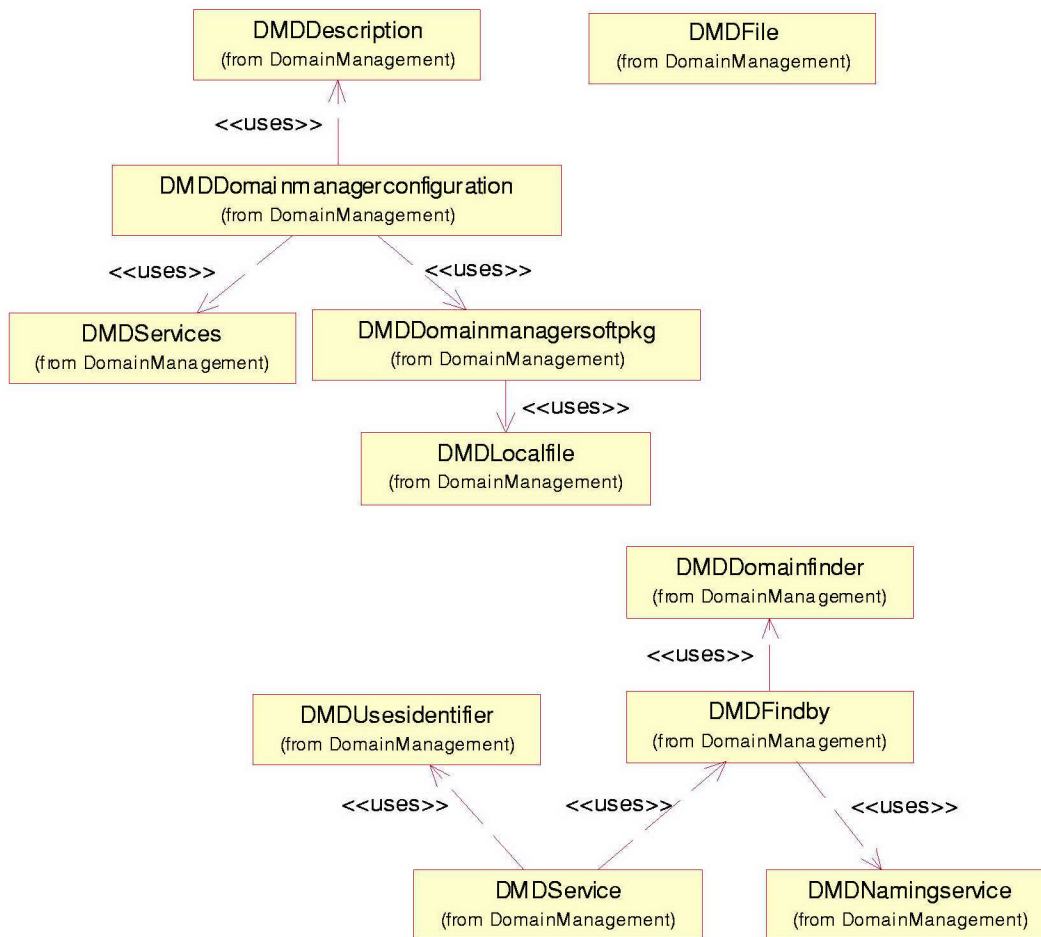


Figure 10-18: Class Diagram of DMD XML Business Objects

DPD XML BUSINESS OBJECTS

The DPD XML business objects are derived from the SCA Device Package Descriptor (DPD) DTD. These classes are used to store and provide access to XML data with a DPD XML file. Each class represents a corresponding element definition inside the DPD DTD.

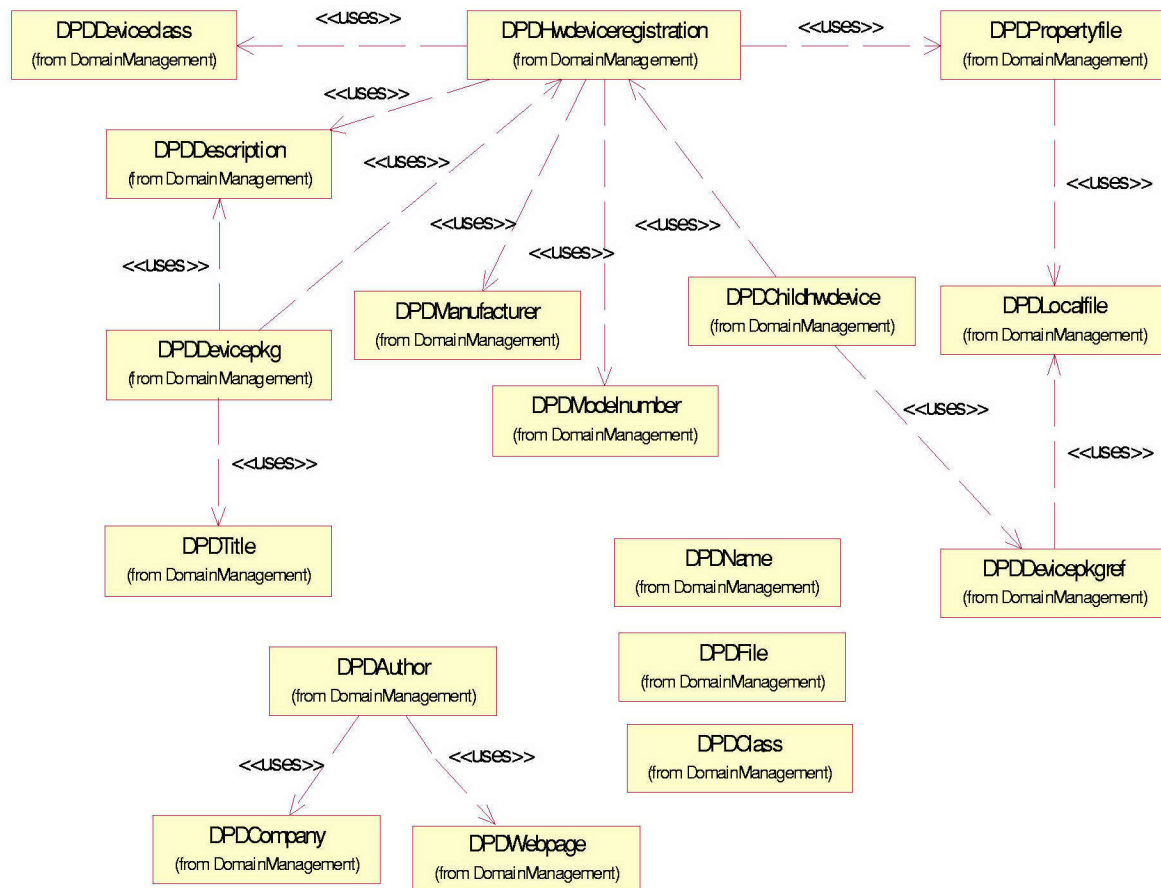


Figure 10-19: Class Diagram of DPD XML Business Objects

PRF XML Business Objects

The PRF XML business objects are derived from the SCA Properties (PRF) DTD. These classes are used to store and provide access to XML data with a PRF XML file. Each class represents a corresponding element definition inside the PRF DTD.

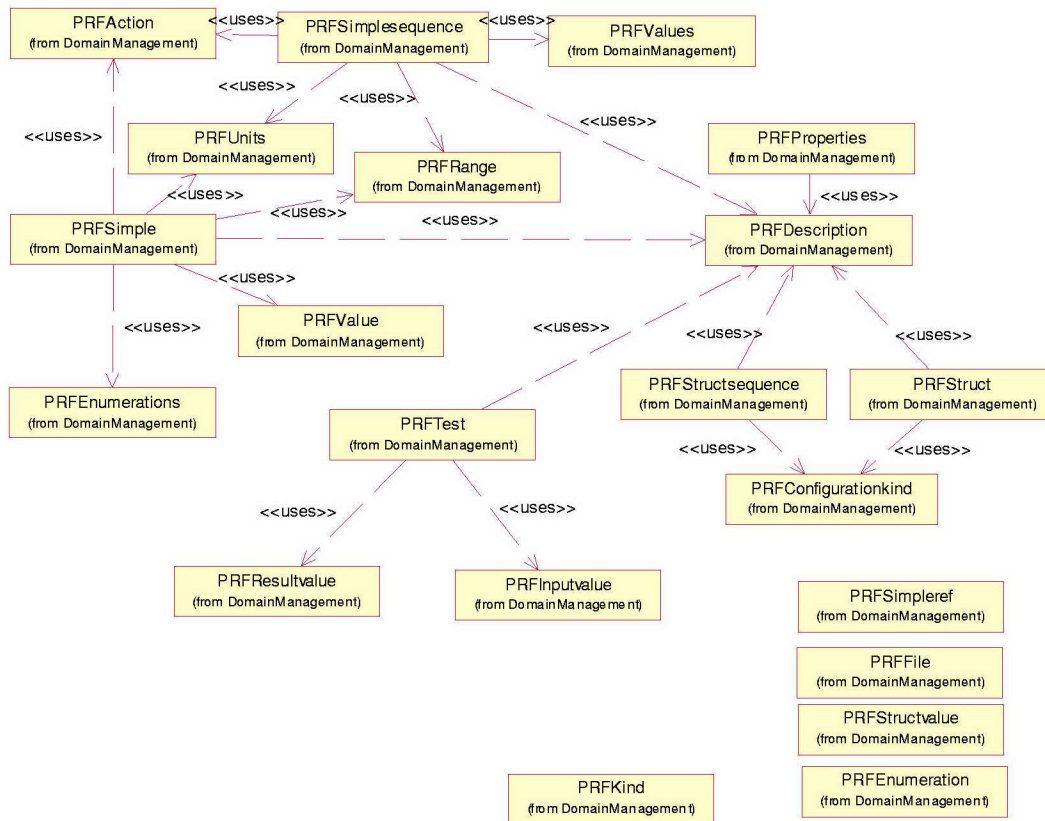


Figure 10-20: Class Diagram of PRF XML Business Objects

Profile XML Business Objects

The Profile XML business objects are derived from the SCA Profile DTD. These classes are used to store and provide access to XML data with a Profile XML file. Each class represents a corresponding element definition inside the Profile DTD.

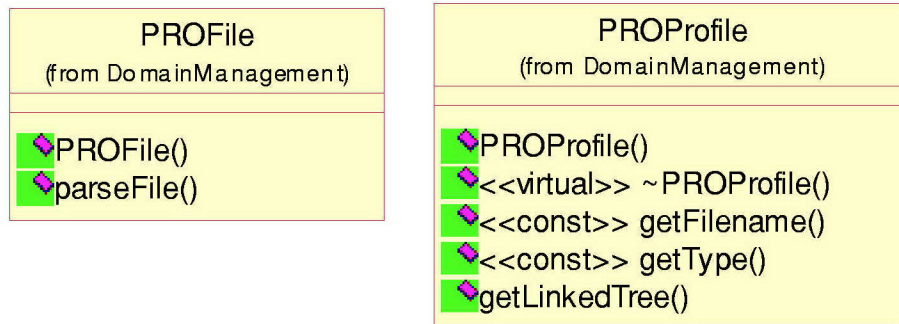


Figure 10-21: Class Diagram of Profile XML Business Objects

SAD XML BUSINESS OBJECTS

The SAD XML business objects are derived from the SCA Software Assembly Descriptor (SAD) DTD. These classes are used to store and provide access to XML data with a SAD XML file. Each class represents a corresponding element definition inside the SAD DTD.

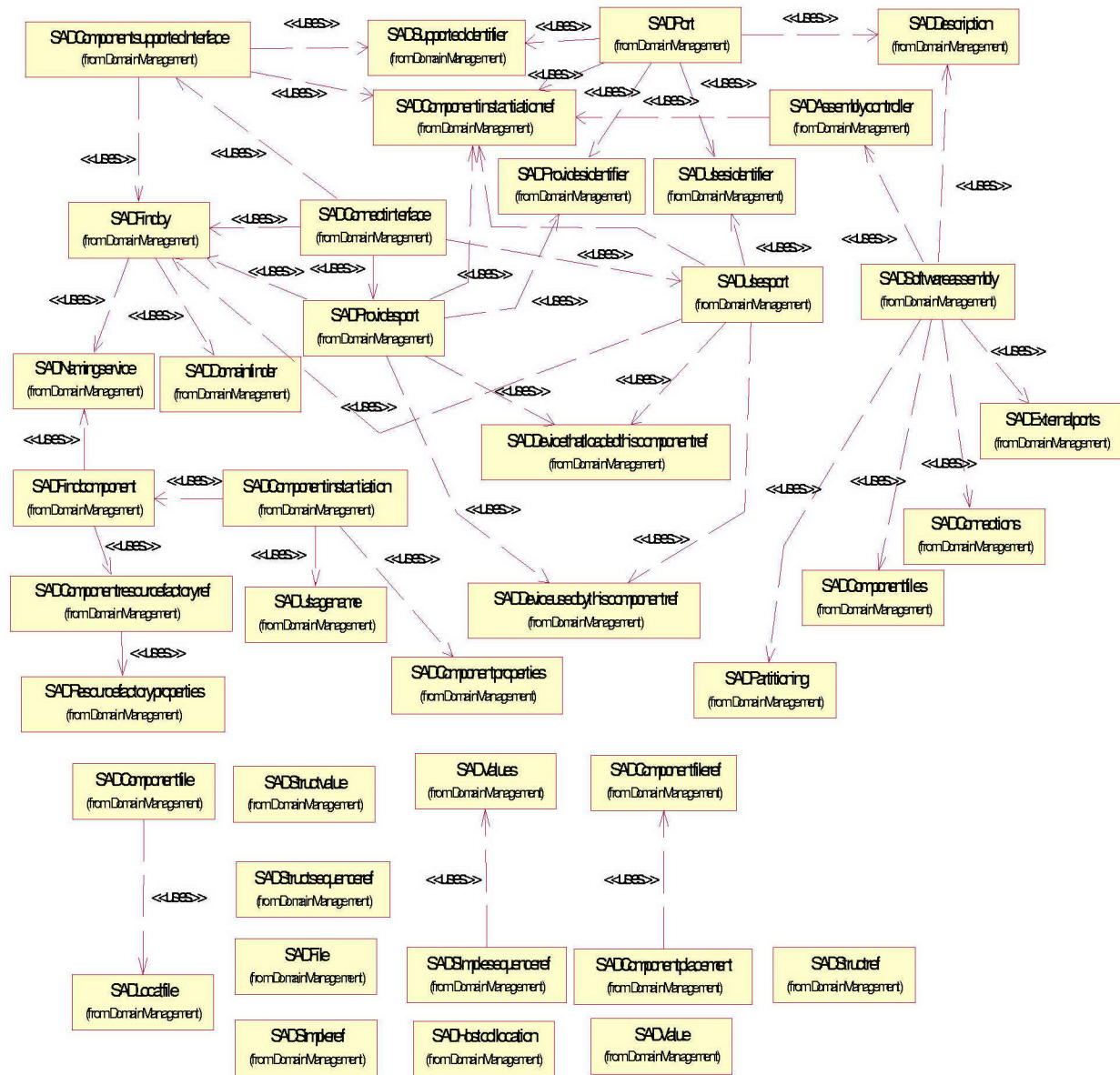


Figure 10-22: Class Diagram of SAD XML Business Objects

SCD XML BUSINESS OBJECTS

The SCD XML business objects are derived from the SCA Software Component Descriptor (SCD) DTD. These classes are used to store and provide access to XML data with a SCD XML file. Each class represents a corresponding element definition inside the SCD DTD.

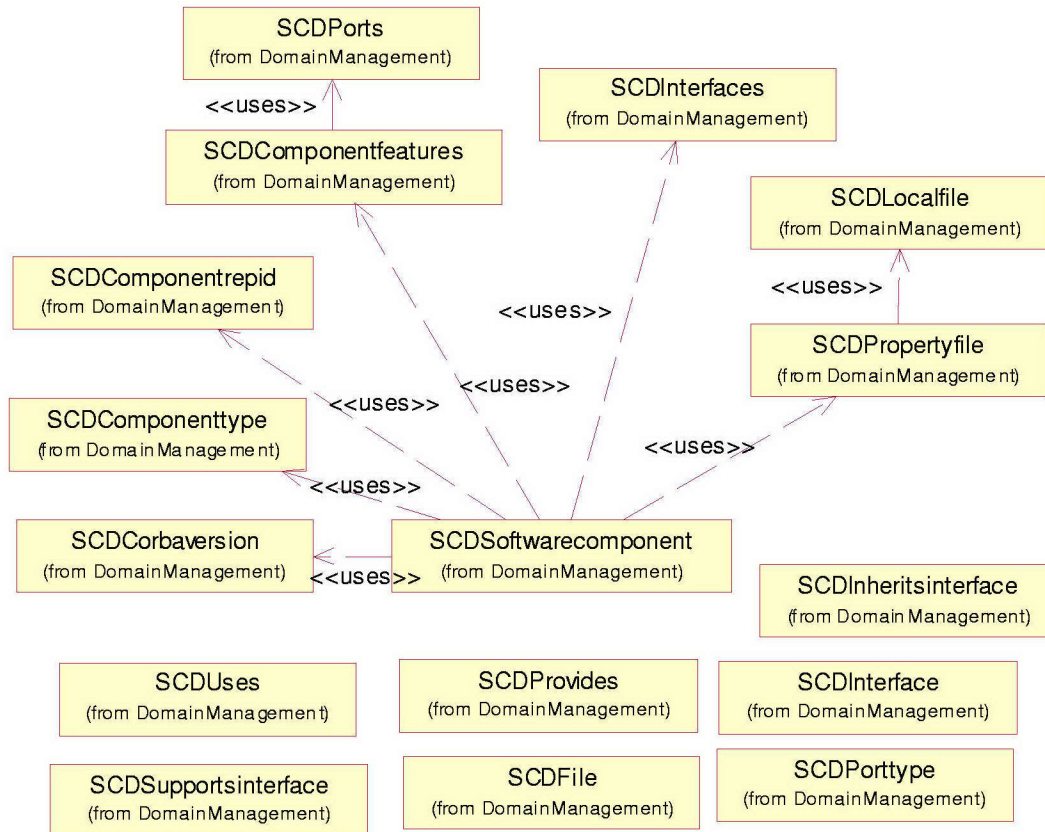


Figure 10-23: Class Diagram of SCD XML Business Objects

SPD XML BUSINESS OBJECTS

The SPD XML business objects are derived from the SCA Software Package Descriptor (SPD) DTD. These classes are used to store and provide access to XML data with a SPD XML file. Each class represents a corresponding element definition inside the SPD DTD.

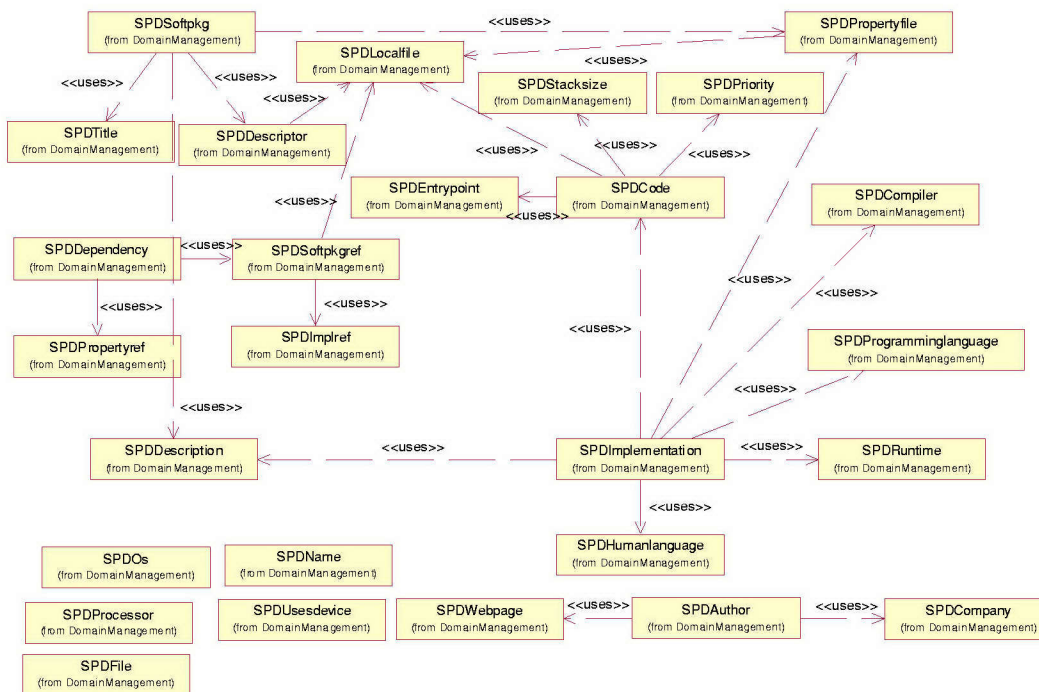


Figure 10-24: Class Diagram of SPD XML Business Objects

Node Booter

The Node Booter is an executable component implementing the Core Framework interfaces for a Device Manager servant object. The class diagrams for the Device Manager can be seen in Figure 10-25.

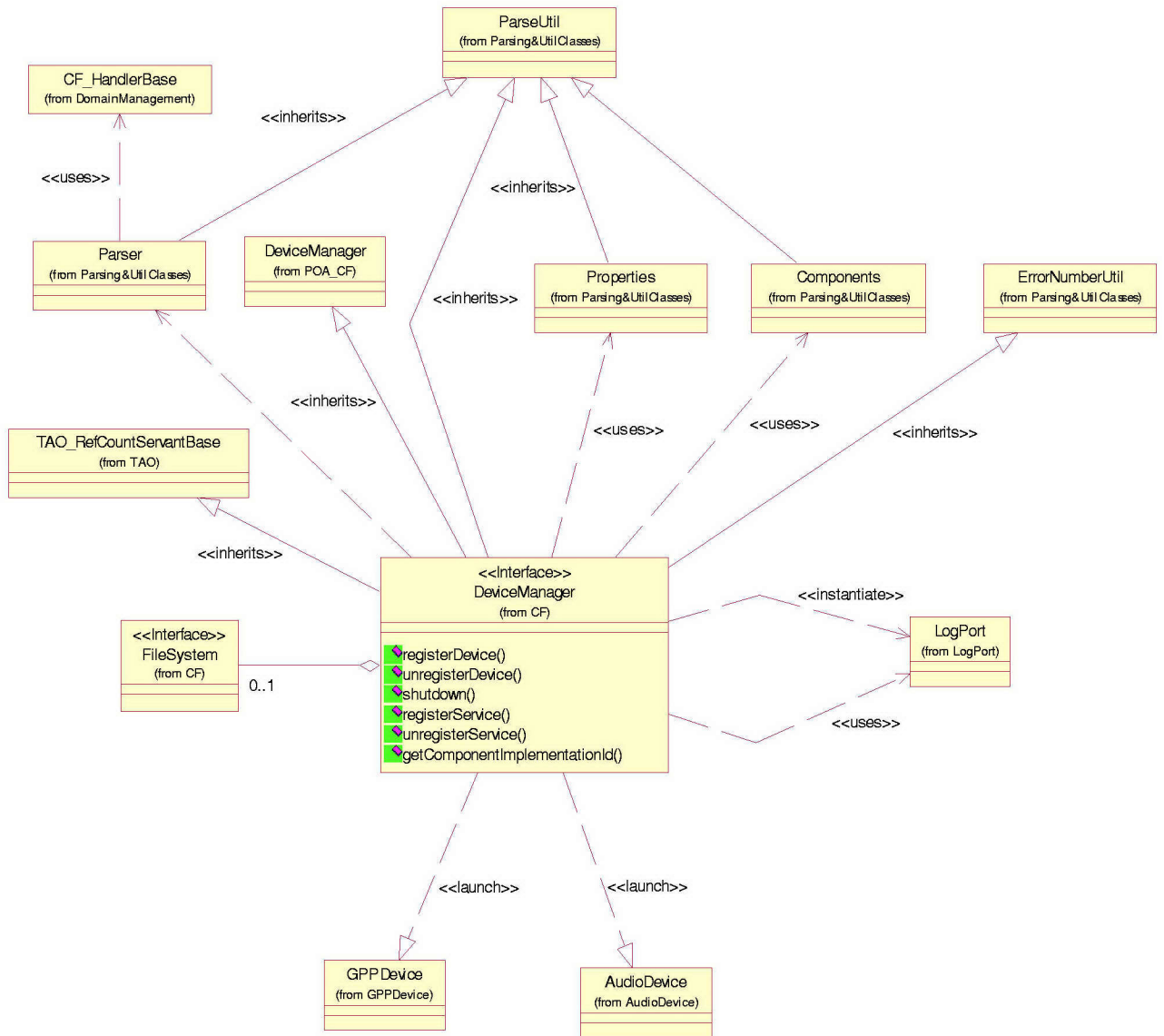


Figure 10-25: Device Manager Class Diagram

The Node Booter component also has internal and external dependencies to ACE, TAO, Naming Service, Event Service, SCA Interface, File Service, Log Port, Audio Device and GPP Device.

HW Interface

The HW Interface is a shared library component providing the client stubs and server skeletons for the interfaces (APIs) used for data communication with OrcaCF device components. These interfaces are defined in IDL and contained in the SimpleWaveform.idl file. All of the C++ code contained within this component has been auto-generated by running the IDL file through the TAO IDL compiler. These interfaces can be seen in Figure 10-26.

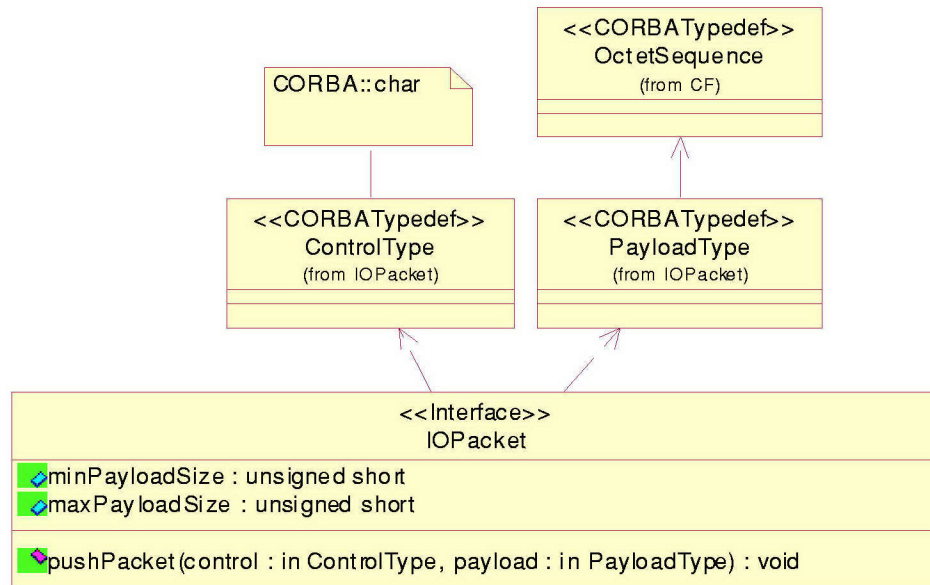


Figure 10-26: HW Interface Class Diagram

The HW Interface component has been developed as a shared library so that it can be dynamically loaded and shared between OrcaCF components. For example, the Sound Demo Waveform Application utilizes this library to exchange PCM data with the Audio Device.

The HW Interface component also has external dependencies to ACE and TAO

Audio Device

The Audio Device is a shared library component implementing the Core Framework interfaces for a Device servant object and an IO Packet servant object. The class diagram for the Audio Device can be seen in Figure 10-27.

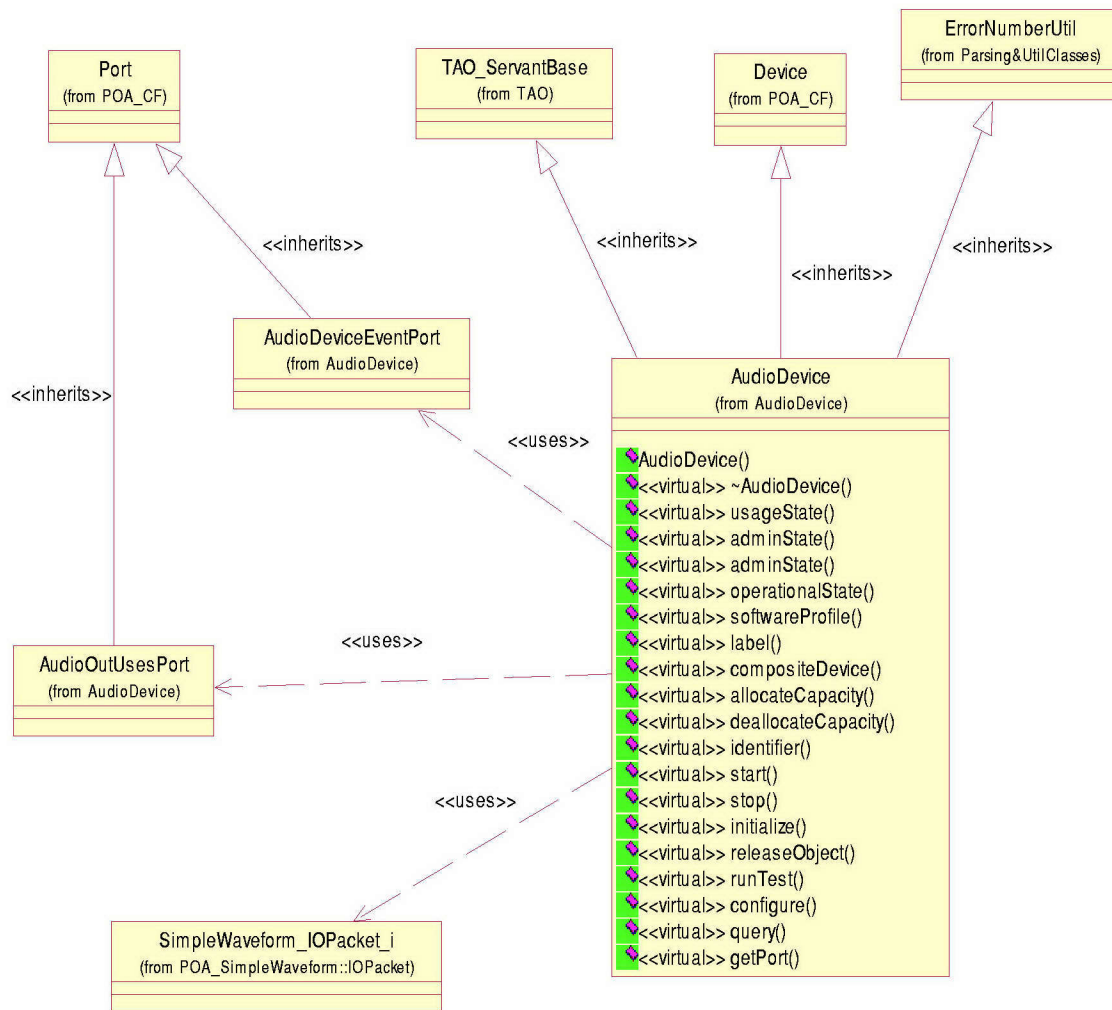


Figure 10-27: Class Diagram for Audio Device

The Audio Device has been implemented as a shared library for efficiency reasons. CORBA calls between objects running in the same process are much more efficient than CORBA calls between processes. Since the Audio Device is instantiated within the same process as the Device Manager and GPP Device, calls between these objects are more efficient than if they were developed as executables.

The Audio Device has been built around the Open Sound System (OSS). OSS is an open system audio architecture that supports off-the-shelf audio hardware. The operating system and sound device must support OSS in order for the Audio Device to function properly.

The Audio Device component also has internal and external dependencies to ACE, TAO, Naming Service, SCA Interface and HW Interface.

GPP Device

The GPP Device is a shared library component implementing the Core Framework interfaces for an Executable Device servant object. The class diagram for the GPP Device can be seen in Figure 10-28.

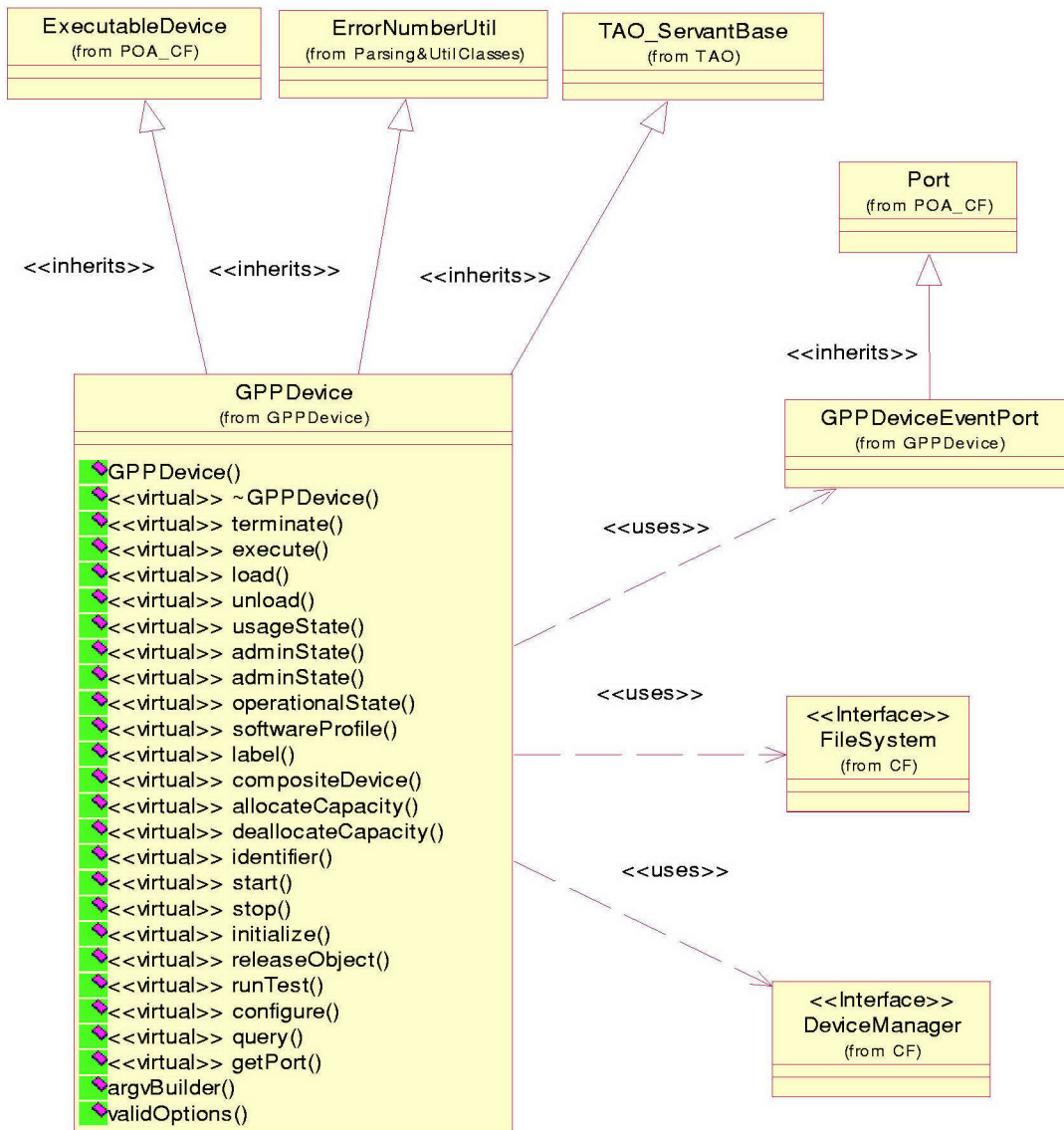


Figure 10-28 GPP Device Class Diagram

The GPP Device has been implemented as a shared library for efficiency reasons. CORBA calls between objects running in the same process are much more efficient than CORBA calls between processes. Since the GPP Device is instantiated within the same process as the Device Manager and Audio Device, calls between these objects are more efficient than if they were developed as executables.

The GPP Device component also has internal and external dependencies to ACE, TAO and SCA Interface.

Sound Demo Waveform Application

In order to test and demonstrate the OrcaCF, a sample waveform application has been developed. The waveform application developed for this purpose was the Sound Demo Waveform Application.

The Sound Demo Waveform Application consists of two packages that represent its components. These packages are the Sound Demo Assembly Controller and the Sound Demo Resource.

Sound Demo Assembly Controller Package

The Sound Demo Assembly Controller is an executable component implementing the Core Framework interfaces for a Resource servant object. The class diagram for the Sound Demo Assembly Controller can be seen in Figure 10-29. The Application Factory loads and launches the Sound Demo Assembly Controller on the GPP Device. The Application Factory also configures and makes connections to the Sound Demo Assembly Controller based on the Sound Demo SAD.

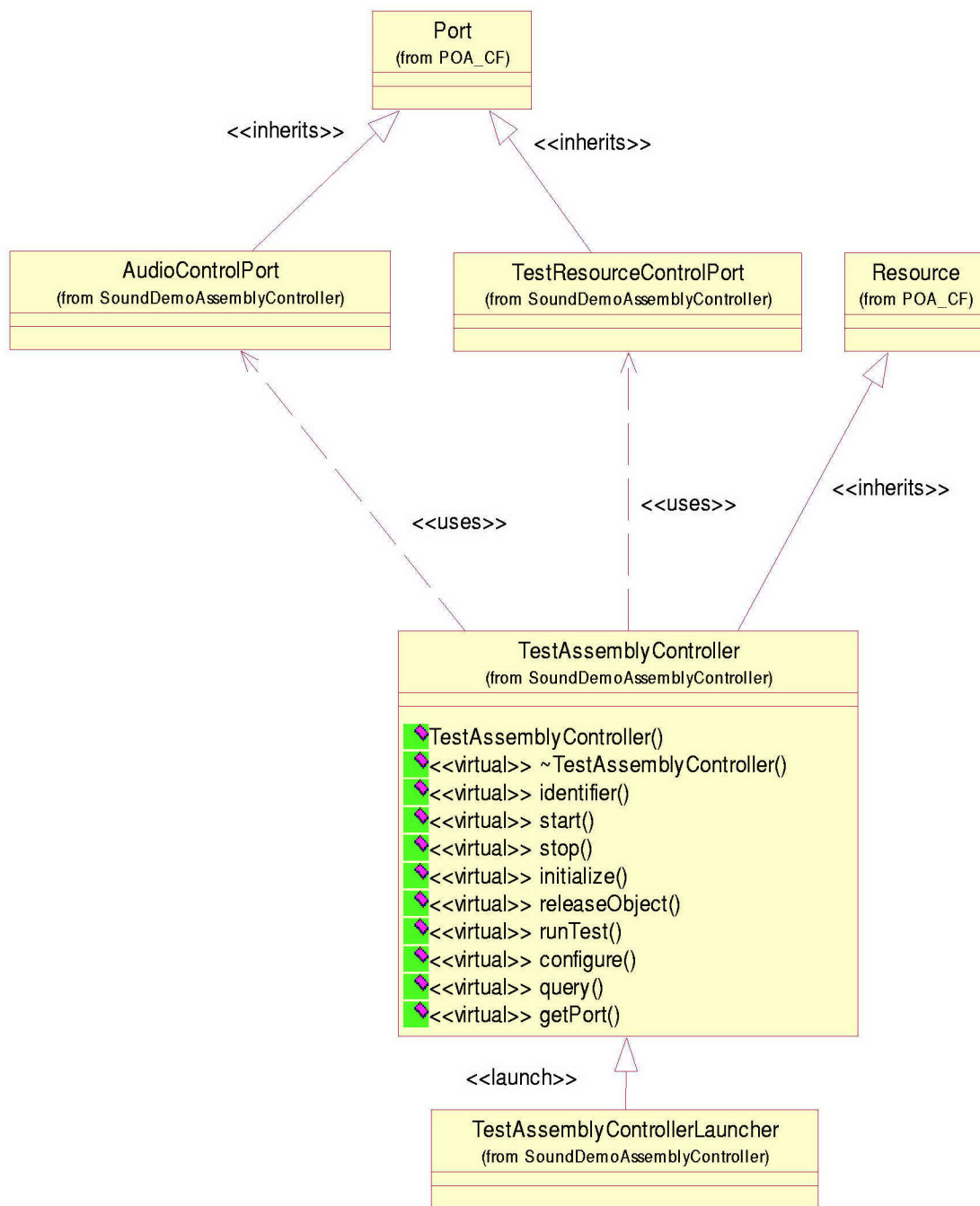


Figure 10-29 Sound Demo Assembly Controller Class Diagram

The Sound Demo Assembly Controller manages the Sound Demo application by controlling the Sound Demo Resource and Audio Device that it is connected to as defined in the Sound Demo SAD. The Sound Demo Assembly Controller is also the assembly controller for the Sound Demo as defined in the SAD.

The Sound Demo Assembly Controller has internal and external dependencies to ACE, TAO, Naming Service and SCA Interface.

Sound Demo Resource Package

The Sound Demo Resource is an executable component implementing the Core Framework interfaces for a Resource servant object. The class diagram for the Sound Demo Resource can be seen in Figure 10-30. The Application Factory loads and launches the Sound Demo Assembly Controller on the GPP Device. The Application Factory also configures and makes connections to the Sound Demo Resource based on the Sound Demo profile.

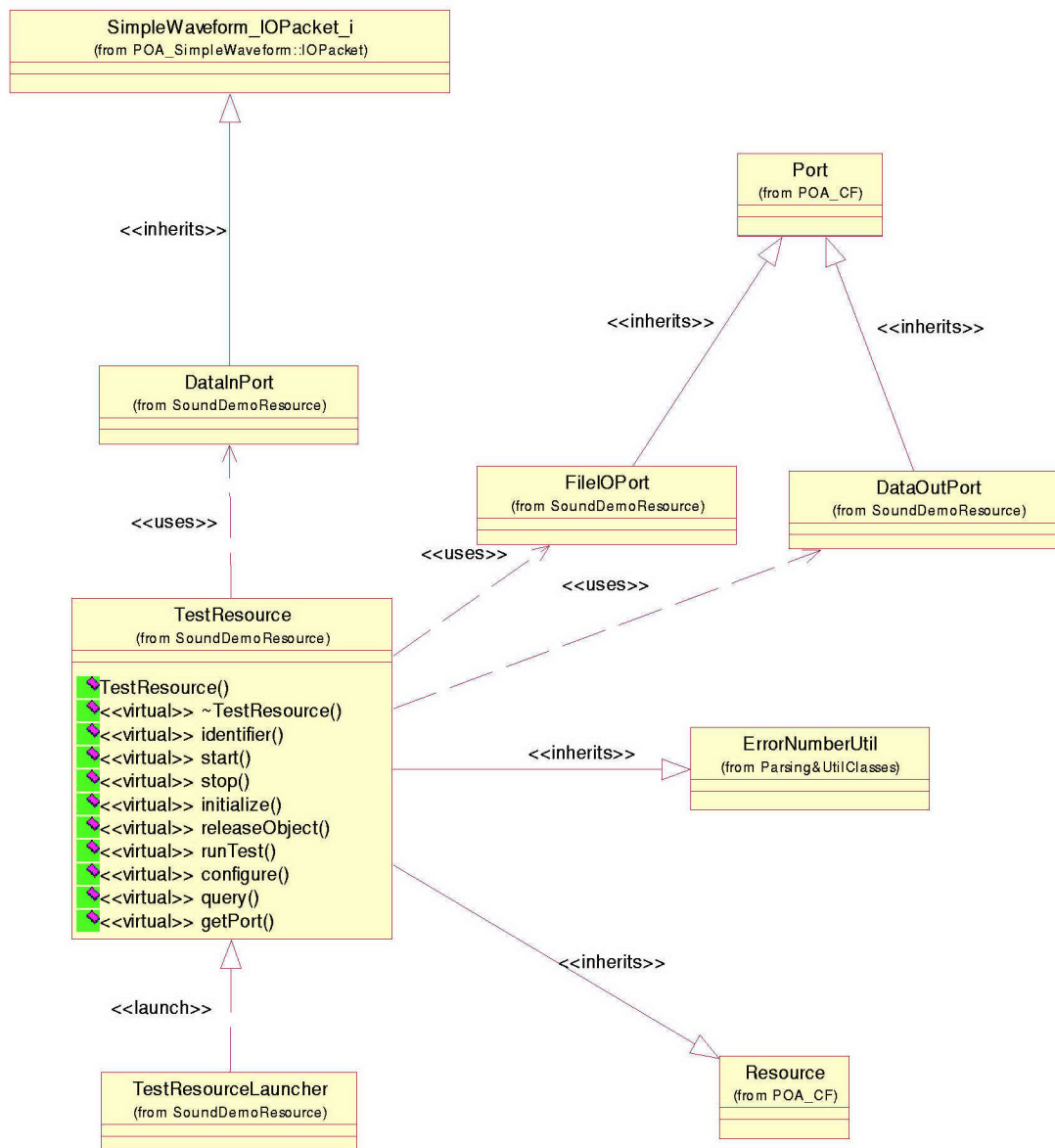


Figure 10-30 Sound Demo Resource Class Diagram

The Sound Demo Resource is connected to the Audio Device, File Service and Sound Demo Assembly Controller. The Sound Demo Assembly Controller controls the Sound Demo Resource by utilizing the `start()`, `stop()` and `configure()` interface functions. In record mode, the Sound Demo Resource receives PCM data from the Audio Device on its **DataInPort** and writes the data out to a file using the File Service through its **FileIOPort**. In play mode, the Sound Demo Resource reads data from a file using the File Service through its **FileIOPort** and sends it to the Audio Device through its **DataOutPort** for playback.

The Sound Demo Resource has internal and external dependencies to ACE, TAO, Naming Service and SCA Interface.

Application HMI

The Application HMI is an executable component used to launch and control waveform applications on the OrcaCF. The Application HMI obtains a reference to the Domain Manager through the Naming Service. It then obtains a list of installed applications from the Domain Manager utilizing the `applicationFactories()` attribute of the Domain Manager. The list of installed applications is presented to the user for selection. Upon the user selecting an application to create the Application HMI calls `create()` on the appropriate Application Factory, which returns a reference to the Application interface of the create application. The Application HMI then controls the waveform application by calling the `start()`, `stop()`, `configure()` and `releaseObject()` functions on the Application interface.

The Application HMI has internal and external dependencies to ACE, TAO, Naming Service and SCA Interface.

Event Viewer

The Event Viewer is an executable component used to view events generated on the ODM Event Channel and IDM Event Channel. The Event Viewer obtains a reference to the Domain Manager through the Naming Service. It then connects to the event channels using the `registerWithEventChannel()` function of the Domain Manager. As events are broadcast on the ODM or IDM Event Channels, the Event Viewer displays the messages to the screen. The class diagram of the Event Viewer can be seen in Figure 10-31.

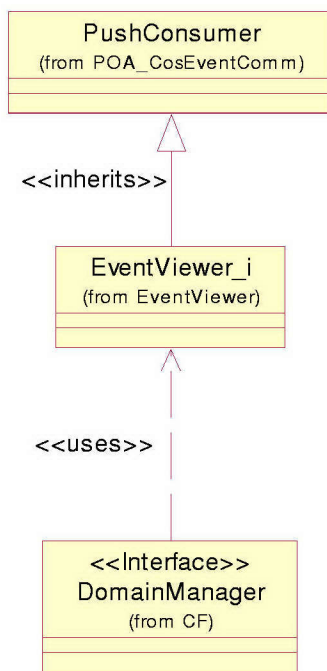


Figure 10-31 EventViewer Class Diagram

The Event Viewer has dependencies on ACE, TAO, Naming Service, Event Service and SCA Interface.

Log Service

The Log Service is an executable component implementing the Core Framework interfaces for a Log servant object. The class diagram for the Log Service can be seen in Figure 10-32. The Log Service is

executed on the GPP Device by the Device Manager as defined in the Node Booter's DCD. Connections to the Log Service are defined in the OrcaCF profile in the DCD, DMD and SAD. Connections to the Log Service are made via the Log Port component.

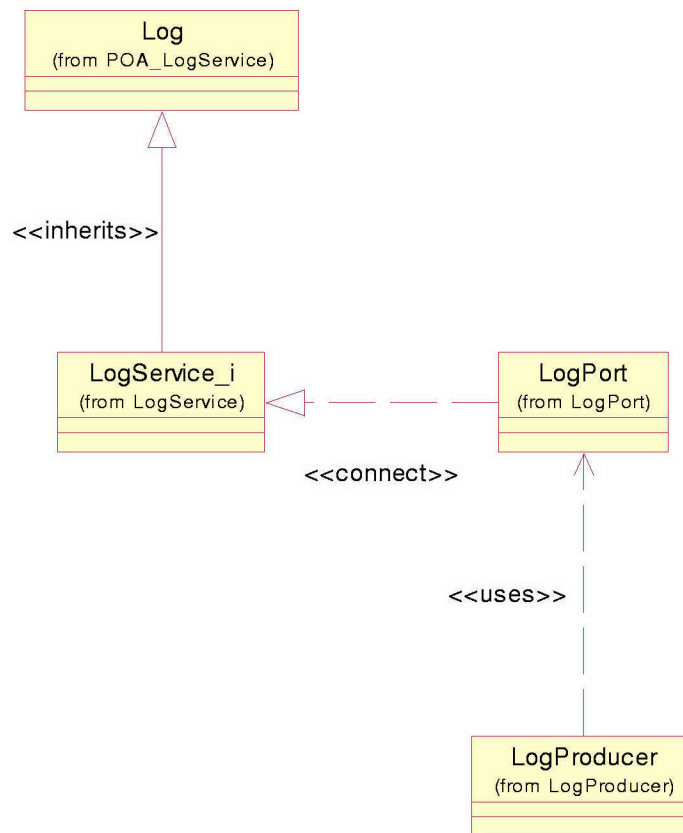


Figure 10-32 LogService Class Diagram

The Log Service has internal and external dependencies to ACE, TAO and SCA Interface.

Log Viewer

The Event Viewer is an executable component used to view events generated on the ODM Event Channel and IDM Event Channel. The Event Viewer obtains a reference to the Domain Manager through the Naming Service. It then connects to the event channels using the `registerWithEventChannel()` function of the Domain Manager. As events are broadcast of the ODM or IDM Event Channels, the Event Viewer displays the messages to the screen. The class diagram of the Event Viewer can be seen in Figure 10-31. The Log Viewer is an executable component used to view log records within the OrcaCF Log Service. The Log Viewer obtains a reference to Log Service through the Device Manager's `registeredServices()` attribute. The Log Viewer obtains a reference to the Device Manager through the Domain Manager's `deviceManagers()` attribute, and obtains a reference to the Domain Manager through the Naming Service. The Log Viewer has internal and external dependencies to ACE, TAO, Naming Service and SCA Interface.

Appendix E. Errata

Scope

This section addresses implementation decisions on pending SCA change proposals. The SCA v2.2 specification has some discrepancies among the text of the main document, the IDL in SCA Appendix C, and the XML in SCA Appendix D. These discrepancies are addressed in various change proposals. In order to implement a Core Framework (CF) consistent with SCA v2.2, developers must choose a design approach to resolve the discrepancies in the SCA specification. These design decisions are referred to as “Errata” because they are associated with errata changes that must be made to the SCA v2.2 specification. This section also includes limitations of this version of the Core Framework (OrcaCF v1.1.0) that are important for developers planning to work with this software.

Errata

IDL

There are several discrepancies between the SCA version 2.2 and the IDL provided in SCA Appendix C. These differences are noted below along with the implementation used for the development of the OrcaCF. OrcaCF was implemented using the IDL as presented in SCA Appendix C as precedence over the text of the SCA specification.

1. The LogService::LogLevelType enumeration as defined in section 3.1.2.3.2.1.1 of the SCA lists SECURITY_ALARM as the first element and FAILURE_ALARM as the second element of the enumeration. The IDL provided for the LogService does not list SECURITY_ALARM in the LogLevelType and FAILURE_ALARM is listed as the first element of the enumeration. For this discrepancy the OrcaCF used the LogService IDL, and therefore the log service does not allow for the setting of a SECURITY_ALARM level type, and the entire enumeration is off by a count of one from the SCA specification.
2. The SCA section 3.1.3.2.6.3.6 has defined the sequence of LogLevelType as LogLevelSequence, but the LogService IDL has defined the sequence of LogLevelType as LogLevelSequenceType. The OrcaCF has implemented the LogService using the LogService::LogLevelSequenceType as defined in the LogService IDL.
3. The CF::DeviceManager::unregisterService operation as defined in section 3.1.3.2.3.6.8.2 of the SCA has a parameter named unregisteringService, while in the IDL it is defined as unregisteredService. The OrcaCF has implemented the unregisterService operation using the unregisteredService parameter.
4. The CF::DeviceManager::unregisterService operation as defined in section 3.1.3.2.3.6.8.2 of the SCA has the exception UnregisterError, the IDL does not have this exception. The OrcaCF has implemented the unregisterService operation without the UnregisterError exception.
5. The SCA section 3.1.3.2.6.3.6 has defined the constant CF::ExecutableDevice::STACK_SIZE_ID. The IDL has defined this constant as CF::ExecutableDevice::STACK_SIZE. The OrcaCF has been implemented using the constant CF::ExecutableDevice::STACK_SIZE as defined in the IDL.

6. The CF::DomainManager::registerService operation as defined in section 3.1.3.2.3.6.7.2 of the SCA does not include the exception CF::InvalidProfile, the IDL includes this exception with the operation. The OrcaCF has been implemented using the operation CF::DomainManager::registerService as it is defined in the IDL with the CF::InvalidProfile exception included.
7. The OrcaCF could not be implemented with the CF::ErrorNumberType as defined in section 3.1.3.5.13 of the SCA due to conflicts with the Linux OE POSIX error number type. The code will not compile properly using the error number types as defined in the SCA. To avoid this conflict, the OrcaCF has been implemented with the string “CF_” appended at the beginning of each of the CF::ErrorNumberType definitions.

XML

There are several interpretation issues and validation issues concerning the XML as defined in SCA 2.2 Appendix D, and with the DTD files. This section provides an explanation regarding the changes made to the XML in order to implement the OrcaCF. This section also points out important OrcaCF XML limitations for developers planning to write their own XML and software, for use with the OrcaCF.

8. The Device Configuration Descriptor as defined in SCA 2.2 Appendix D Attachment 1 has defined the element “componentinstantiationref” twice. The second definition has been removed from the OrcaCF DTD.
9. The Device Package Descriptor as defined in SCA 2.2 Appendix D Attachment 1 has defined the element “description” twice. The second definition has been removed from the OrcaCF DTD.
10. The Domain Manager Configuration Descriptor as defined in SCA 2.2 Appendix D Attachment 1 has defined the element “devicemanagersoftpkg” instead of “domainmanagersoftpkg.” The element “devicemanagersoftpkg” has been replaced with “domainmanagersoftpkg” in the OrcaCF DTD.
11. The Software Package Descriptor as defined in SCA 2.2 Appendix D Attachment 1 has defined the element “propertyfile” twice. The second definition has been removed from the OrcaCF DTD.
12. There are some interpretation differences between the OrcaCF and the JTAP tool (July 03 version) with regards to the XML as defined in the SCA. For additional information regarding these issues refer to Appendix F of the OrcaCF Software Users Manual.
13. The SCA allows many variations and combinations with the XML files for the Domain Profile. The OrcaCF does not support all possible variations of XML. It has only implemented the XML parsing necessary to accommodate the SoundDemo waveform application and devices delivered with the OrcaCF. However, this should be sufficient for most applications. For additional information regarding the level of XML parsing within the OrcaCF, refer to Appendix F of the OrcaCF Software Users Manual.
14. DCD XML file: <filesystemnames> - The DeviceManager does not currently handle multiple FileSystems. We have a generic FileSystem that uses the environment variable

“ORCACF_ROOT” as its filesystem name. The optional <filesystemnames> element should not be in the DCD XML file.

15. DCD XML file: <componentinstantiation> - The DeviceManager does not currently handle multiple component instantiations within a single <componentplacement> element. Multiple instantiations can be accomplished by creating multiple <componentplacement> elements with single <componentinstantiations> elements. Each instantiation must have a unique UUID.
16. DCD XML file: <componentproperties> - The DeviceManager does not currently handle the optional component properties listed under <componentinstantiation> elements. In order to specify properties for a component, you must list a PRF XML file in the component's SPD XML file.
17. PRF XML file(s): The DeviceManager currently only handles configure properties of type <simple> and <simplesequence>. All data types for the <simple> element are handled, but only ulong is handled for the <simplesequence> element.
18. SPD XML file(s): <dependency> - The DeviceManager currently does not handle the dependency element.
19. SPD XML file(s): <runtime> - The DeviceManager currently does not handle the runtime element.
20. SPD XML file(s): <code> - The DeviceManager currently loads and executes EXECUTABLES only listed under the <code> element. It does not handle SHARED LIBRARIES with entryptoints. A <localfile> and <entryptoint> are REQUIRED for any Device or Service that is listed in the DCD under <componentplacement>.
21. SPD XML file(s): <stacksize><priority> - The DeviceManager does not currently handle these two options. Both of these options are unstable and it is highly recommended they NOT be used.
22. DCD XML file: Our GPPDevice MUST be listed in the DCD XML file, even if other developers provide their own CF::ExecutableDevice. The GPPDevice is a CF::ExecutableDevice that is used by our DeviceManager to launch/execute other Devices or Services listed in the DCD. If a Device or Service listed in the DCD does not specify a <deployondevice> element, the DeviceManager uses the GPPDevice as the default CF::ExecutableDevice.

SOURCE

This section points out important limitations in the implementation of the OrcaCF v1.1.0. This information is important for developers planning to write their own software, for use with the OrcaCF.

DomainManager

The registerDevice() function of the DomainManager currently does not parse a registering device's XML to check for a valid profile.

The Domain Manager does not currently store connections listed within the DCD as pending connections. The DomainManager assumes that when the DeviceManager has registered with the DomainManager, that all of the DeviceManager's devices and services have already registered back with it. If a third-party DeviceManager registers with the DomainManager prior to all device and service registration, exceptions may be thrown. This problem effects the registerDeviceManager(), registerDevice(), and registerService() functions.

The DomainManager does not currently store connections listed within an SAD as pending connections. The DomainManager and ApplicationFactory assume that all necessary devices and services needed for an application are running and registered prior to creation of an application.

The DomainManager only parses the <simple> properties, and <simplesequence> properties of type along. This effects the query() and configure() functions.

ApplicationFactory

The ApplicationFactory assumes that all devices and services necessary for application creation are already registered with the DomainManager. If the SAD lists a service that is not yet registered with the DomainManager, the ApplicationFactory will throw an exception.

The ApplicationFactory only parses <simple> and <simplesequence> properties. All other properties are ignored by the ApplicationFactory.

File Service

Due to the current configuration of the CORBA middleware, the OrcaCF is unable to throw exceptions and provide return values at the same time. The OrcaCF assumes that if an exception is caught the return value is invalid and should not be touched. This effects the open() and create() functions within the FileManager and FileSystem. This problem also affects other SCA defined functions where an exception and return value are both specified.

When copying a directory the OrcaCF File Service only copies the directory and not the contents of the directory.

DeviceManager :

- `shutdown()` : The DeviceManager's shutdown() operation is currently unstable. There are ORB concurrency issues that have not been resolved at this time. It is highly recommended that you do not invoke the shutdown() operation.

GPPDevice :

- `releaseObject()` : The GPPDevice's releaseObject() operation is currently unstable. It is highly recommended that you do not invoke the releaseObject () operation at this time.

`execute()`: The GPPDevice only executes EXECUTABLE files. Passing in a function name is not implemented. EXECUTABLES are launched as new "processes" as opposed to "threads." The GPPDevice handles the STACK SIZE and PRIORITY being passed in as options, but the PRIORITY will not actually be SET because it requires root privileges and is unstable. The STACK SIZE option is functional, but it is also unstable and we recommend not using it.

Appendix F. Release Notes

Scope

The OrcaCF v1.1.0 is an initial release.

Overview

The OrcaCF implements the Joint Tactical Radio System (JTRS) Software Communications Architecture (SCA) version 2.2 specification. It was developed in C++ and uses ACE/TAO for the CORBA middleware, Xerces for the XML parser, and Linux for the Operating System.

The OrcaCF v1.1.0 is ideal for rapid prototyping of waveforms built to SCA specifications since it is PC-based, and uses Open Source software components. It has been built and tested on Red Hat 7.3 (gcc 2.9.x), RH 9.0 (gcc 3.2.x), and Fedora Core 1 (gcc 3.3.x).

The OrcaCF runs on a standard Linux PC and comes with a simple audio recorder Sound-Demo "waveform" that is used to demonstrate the capabilities of the OrcaCF. The Sound-Demo application consists of an Audio Device, Recorder Resource, Assembly Controller, and a simple Human Machine Interface (HMI).

For more information, see www.OrcaCF.com.

Release Notes

Change Proposals:

This release includes change proposals (CPs) that have been incorporated into the latest SCA Spec v2.2.1. The implemented CPs are: 13, 15, 26, 44, 45, 70, 73, 74.

Appendix G. Code Style Guide

Scope

This document describes a C++ programming style guide to be used for the development of C++ Software applications for the Joint Tactical Radio System (JTRS) Software Communications Architecture (SCA). For information on the JTRS SCA, refer to <http://jtrs.army.mil>.

Objectives

The objectives of this style guide are to:

- Standardize source code within the project;
- Provide source code readability between developers;
- Increase code understanding within the development team; and
- Provide a basis for formal inspection.

To produce robust software, formal inspection of developed source code is a necessity. Formal inspection requires that several software engineers closely scrutinize a colleague's source code, report major and minor defects, and agree as to whether or not the source code performs the required work. All Software Communication Architecture (SCA) development done by L-3 Communications Government Services Inc. (L3) and its subcontractors shall use this programming style guide to aid in this effort.

Due to the environment associated with SCA development, some of the following rules may differ slightly from standard C++ styles. This is due to the introduction of Common Object Request Broker Architecture (CORBA) into the development. Much of the code has been auto generated by the Object Request Broker (ORB) compiler and the style has been defined to remain consistent with the auto generation features of the Adaptive Communication Environment/The Ace ORB (ACE/TAO) ORB, the ORB being used by L3 for SCA development. The L3 development team followed ACE/TAO and CORBA styles for consistency.

Document Overview

This document provides general guidance as it relates to programming style. This document does not cover design or technique, but instead provides a programming style to ensure consistent and readable source code. This Style Guide is used by the L3 development team as a reference while programming JTRS applications.

The remaining sections of this document are organized in the following manner:

Section 0 is a list of reference documents for this Style Guide.

Section 0 is a set of rules for documenting the source code.

Section 0 provides rules for applying white space to the source code.

Section 0 is a set of general naming conventions used on this project.

Section 0 provides guidelines on the declaration and initialization of variables.

Section 0 is a set of general rules regarding preprocessor directives.

Section 0 provides general formatting rules for writing statements.

Section 0 is a list of defect avoidance guidelines.

Section 0 is a code checklist which will be used in conjunction with sections 2.0 to 9.0 to review code.

Relationship to Other Documents

This Style Guide provides a standard set of style for coding the OrcaCF. For additional guidelines and lessons learned, please refer to SUM Appendix B - Developers Notes.

Referenced Documents

- JMCIS MARITIME COMMAND INFORMATION SYSTEM (JMCIS) Re-Engineering C++ Programming Style Guide, 16 December 1999

DOCUMENTATION

There are two major objectives behind the standardization of the documentation:

1. Readability that is standardized to the SCA
2. Provide for the generation of code count metrics.

In order to accommodate the recording of metrics during the development of the OrcaCF, it is important to document all source code in a standard manner. These documentation standards are strictly followed to ensure accuracy of the metrics, and to promote consistency in code development.

Standard Header

All source code files created by L3 use the following header. If required, the comment delineation may be changed to accommodate the development environment. The header used for C++ development appears below:

```
// <Software_Metric_Tag>
/*****
File:      <File name>
Created:   <Date>
Author:    <Author>

*****
*   L-3 Communications GSI               *
*   1300-B Floyd Ave                   *
*   Rome, NY 13440                     *
*   (315) 339-6184                     *
*****

//<license>
//</license>
//<revision>
//</revision>
//<environment>
//</environment>

*****/
```

The markers “<>” identify the items in the header that are to be customized for each source code file. The license, revision, and environment customization tags are used to automate the updating of that information in multiple files through a custom script we developed. This script will copy the desired license, revision, or environment text from a given file, and insert the text between the appropriate start and end markers in the header.

<Software_Metric_Tag> - see section 0 for more information.

<File name> - Shall be the name of the source or header file.

<Date> - Shall be of the format mm/dd/yyyy and will be the date that the file was created.

<Author> - Shall be the name of the primary person responsible for the creation of the file.

<license></license> - inserted between these tags shall be the latest version of the L-3 Communications Government Services Inc. Software License Agreement. See SUM Appendix A for more details on the specific wording of the Software License.

<revision></revision> - inserted between these tags shall be the revision history of the file, which shall be of the following format. The person in charge of configuration management for the overall program shall determine this information.

```
Revision History:
  Version          Date          Comments
  <version>       <date>        <comments>
```

<version> - shall be of the format x.y.z, and shall reflect the version number of the build for the overall program. Numbering shall begin with 1.0.0.

<date> - shall be of the format mm/dd/yyyy and will be the date of the version release for the overall program.

<comments> - shall briefly describe the features/changes/fixes in the release.

<environment></environment> - inserted between these tags shall be the development environment of the file, which shall be of the following format. The Development Environment is the combination of hardware and software that is being used to develop the program.

```
Development Environment:
  Hardware:
    Processor: <processor_type>
    Memory: <memory_amount>
    Peripherals: <peripheral_type>
  Software:
    OS: <operating_system>
    IDE: <development_environment>
    Compiler: <compiler_type>
  Utilities: <add-ons>
  Requirements Met:
    Source: <Source>
```

<processor_type> is the type of processor that the source code is being programmed for.

<memory_amount> is the amount of memory in the development platform.

<peripheral_type> shall be a list of the platform peripherals and their configuration or settings.

<operating_system> is the OS and version of the development platform.

<development_environment> is the Integrated Development Environment (IDE) being used to develop the source code (e.g. KDevelop 2.1).

<compiler_type> is the type of compiler being used to develop the source code (e.g. gcc 2.96-110)

<add-ons> are additional utilities to aid in the development of the source code. These could be CORBA Services, XML parsers, Active-X components, etc. Only the utilities utilized within the source code need to be listed.

Software Metric Tag

Each file, whether auto-generated, reused, or created by L3, shall have a comment on the first line of the file that delineates the type of source code file. This delineator shall have the following structure.

```
// <Software_Metric_Tag>
```

The <Software_Metric_Tag> is used for the purpose of counting Source Lines of Code (SLOC). Productivity is based on an accurate count of new and modified code. Therefore, the Software Metric Tag will be used to separate reused and auto-generated code files from the new and modified source code files. Valid delineators for the Software_Metric_Tag consist of the following:

- new
- reuse
- mod
- auto
- internal

new

The ‘new’ delineator shall identify source code files that are new for the current version of the software product. After the software product is base lined for final version release, the ‘new’ delineator shall be changed to ‘reuse’ in order to designate the file as being created in a previous version of the software product. Source code files in this category are counted for total SLOC and are used for productivity calculations.

reuse

The ‘reuse’ delineator shall identify source code files used “as is” that were not developed for the current version of the software product, but are being delivered with the software product. Reuse code comes from two sources: the first is source code developed for a previous version of the software product. The second source of reuse code comes from third-party development. Source code files in this category are counted for total SLOC, but are not used for productivity calculations.

mod

The ‘mod’ delineator shall identify source code that was previously tagged as ‘reuse’ but has been modified for use within the current version of the software product. The ‘reuse’ tag should only be changed to ‘mod’ if a programmatic change has been made to the source code file. The changing of comments or variable names within the ‘reuse’ file does not warrant a change to ‘mod’. Source code files in this category are counted for total SLOC and are used for productivity calculations.

auto

The ‘auto’ delineator shall identify source code files that were auto-generated by a tool that generates source code. An example of this is an IDL compiler that generates client and servant source code.

Source code files in this category are counted for total SLOC, but are not used for productivity calculations.

internal

The ‘internal’ delineator shall identify source code files that were developed during the current version of the software product, but are not delivered as part of the software product distribution. An example of a source code file that falls into this category is one that contains only unit testing code. Source code files in this category are counted for total SLOC and are used for productivity calculations.

*.h Files

The C++ header files shall be documented in such a way as to describe the performance of the class and functions. The intent of the documentation in the *.h file is to provide enough information in order that the class can be reused in another program based on the *.h file and excluding the *.cpp file.

Functions

All function headings will be documented for readability, starting and ending with a 78 character line of asterisks ‘*’.

Example:

```

/*****
* Function: Destructor ()
* Description: Destroys the CF_FileSystem_I object
* Arguments: None
* Return: None
* Exception: None
* Notes: None
*****/
```

SCA classes and functions

For classes and functions that have been defined in the SCA, the appropriate sections, along with paragraph numbers, shall be cut and pasted into the file just in front of the class or function that it pertains to.

Example: taken from the create() function in the FileSystem implementation file.

```

/*****
* SCA v.2.2
* 3.1.3.3.2.5.5 create.
* 3.1.3.3.2.5.5.1 Brief Rationale.
* The create operation provides the ability to create a new file on the
* FileSystem.
* 3.1.3.3.2.5.5.2 Synopsis.
* File create(in string fileName) raises( InvalidFileName, FileException );
* 3.1.3.3.2.5.5.3 Behavior.
* The create operation shall create a new File based upon the provided
* fileName attribute.
* 3.1.3.3.2.5.5.4 Returns.
* The create operation shall return a File component reference to the
* opened file. The create operation shall return a null file component
* reference if an error occurs.
* 3.1.3.3.2.5.5.5 Exceptions/Errors.
* The create operation shall raise the CF FileException if the file
* already exists or another file error occurred.
* The create operation shall raise the InvalidFileName exception when a
* fileName is not a valid file name or not an absolute pathname.
*****/
virtual ::CF::File_ptr create (
    const char * fileName,
    CORBA::Environment &ACE_TRY_ENV
```

```

)
ACE_THROW_SPEC ((
    CORBA::SystemException,
    CF::InvalidFileName,
    CF::FileException
));

```

Non-SCA Functions

Functions that have not been defined in the SCA that are part of the SCA development shall be documented in a similar manner as the SCA defined functions.

Example:

```

/*****
 * Function: The function name.
 * Description: A description of what the function does.
 * Arguments: A list of the arguments for the function and their purpose.
 * Return: The return values for the function.
 * Exceptions: A list of exceptions that can be thrown by the function.
 * Notes: Any general comments that the programmer feels are necessary.
 *****/

```

*.cpp Files

The *.cpp files shall be documented in order to easily understand the flow of logic used to implement the functions for the class it is associated with. For a description of how the function works and information on using the function, the *.h file will be referred to.

Function Delineators

To allow for the easy location of functions within the *.cpp file, the following example shows the delineation that will be used for each function. The end of the function will also be delineated. After the closing “}” for the function, a comment will be entered as follows: “} // end <functionName>()” The example is taken from the create() function in the FileSystem implementation file.

Example:

```

/*****
 * create()
 *****/
::CF::File_ptr CF_FileSystem_i::create (
    const char * fileName,
    CORBA::Environment &ACE_TRY_ENV
)
ACE_THROW_SPEC ((
    CORBA::SystemException,
    CF::InvalidFileName,
    CF::FileException
))
{
    // implementation source code goes here
} // end create()

```

Branching Statement and Loop Delineators

For readability of code, the end of blocks of code will be delineated clearly for nested blocks and blocks exceeding 10 lines of code. After the closing “}” for the code block, a comment will be entered as follows: “} //end <Block>”. Where Block indicates one of the following: while, if, if/else, switch. Refer to the example in section 0.

WHITE SPACE

White space is included within the source code to enhance the readability of the files. Following are some guidelines regarding the use of white space. All of the subsections refer to the following as an example.

Example:

```
int object::function(  
    int arg1,  
    float arg2)  
{  
    if (expression)  
    {  
        .  
        .  
    }  
    else  
    {  
        switch (c)  
        {  
            case 1:  
                break;  
            case 2:  
                break;  
            default:  
                break;  
        } // end switch (c)  
    } // end if/else  
  
} //end function()
```

Line Length

Maintain a maximum line length of 78 characters for all comment and source lines.

Tabs

Tab characters shall not appear in the source code files. Spaces shall always be used for indentation of lines. For convenience, most editors allow for the conversion of the tabs to spaces.

Indentation Length

The indentation length shall be 3 characters.

Indent Style

Indent bracket and line-up code with bracket.

Statements/Line

There shall only be one statement per line.

Example:

```
int var1 = 0; //Good Practice  
float var2 = 0.0;  
  
int var1 = 0; float var2 = 0.0; //Poor Practice
```

Operators

There shall be at least one space before and after each operator, including the assignment operator.

Example:

```
var = 1 + 5;           //Good Practice  
  
var=1+5;               //Poor Practice
```

Variables

Each variable shall be listed on a separate line.

Example:

```
int var1; }           //Good Practice  
int var2;  
  
int var1,var2;        //Poor Practice
```

Long Statements

In a statement that consists of two or more lines, every line except the first must be indented one additional tab-stop. This is to show that it is a continuation of the first line.

Example:

```
if ((argument1) &&  
    (argument2) &&  
    (argument3))  
{  
    ...  
} // end if
```

Declare Parameter list

List parameters below function.

Example:

```
::CF::File_ptr CF_FileSystem_i::create (  
    const char * fileName,  
    CORBA::Environment &ACE_TRY_ENV  
)
```

Naming Conventions

Names

Choose variable names that suggest the usage.

Class Names

Class names are broken down into three components for the OrcaCF, the prefix, the suffix and the component name. Use upper case letters as word separators in the component name, lower case for the remainder of a word. First character in the component name is upper case.

Example:

```
CF_DomainManager_i.cpp  
prefix_component_name_suffix
```

‘CF_’ is a prefix that signifies a standard Core Framework class.

‘_i’ is suffix that signifies an implementation class.
Class names without a prefix or suffix signify a helper class.

Variables

Variables shall begin with a lowercase letter.

Example:
`string name;`

Multiple Word Names

Where names consist of more than one word, the words are written together and each word that follows the first shall begin with an uppercase letter, with the exception of CORBA suffixes as shown section 0.

Examples:
`mFileName, mFileName_var`

Underscores

Underscores may be used within variable names, between suffix and name, or between prefix and name. Variables will not begin with underscore characters.

Prefix Notation

The identifier for each variable are indicated by the following prefix designations:

- ‘g’ – global variable: a global variables should be used only when absolutely necessary, such as with thread code.
- ‘m’ – member variable: A member variable is a private member variable to a class. These private member variables can be used directly by class member functions, but can only be accessed by non-class member functions through public functions.
- ‘p’ – pointer variable
- ‘o’ – object variable

Pointers shall be prefixed by a ‘p’ with names following starting with uppercase.

Example: `String* pName;`

Globals will be prefixed by a ‘g’ with names following starting with uppercase.

Example: `String gName;`

Combinations of prefixes may be used as necessary.

Example: `String* gpName;`

Functions

The name of a function shall clearly define the action of the function.

Example: `checkForErrors()`, instead of `check()`.

Polymorphic Declarations are permitted

Example: `void RetryValue(int setValue) - set a value`

Enum Names

Enumerations shall be all uppercase with '_' word separators. Enumeration limits should be declared.

Example:

```
enum Msg_T
{
    MSG_MIN,
    MSG_OVERRUN = MSG_MIN, //Minimum of range
    MSG_UNDERRUN,
    MSG_ANOTHER,
    MSG_MAX           //Maximum (last + 1)
};
```

Abstract Data Types, Structures, Typedefs & Enums Names

The names of abstract data types, structures, typedefs, and enumerated types shall begin with an uppercase letter.

General Naming Conventions

Suffix

A name shall be separated from its suffix using an underscore ('_').

TypeNames

TypeNames that differ only by the use of uppercase and lowercase letters shall be avoided.

Abbreviations

Names shall be easily interpreted and understood.

CORBA Suffixes

With the intention of differentiating CORBA objects from C++ objects, the following naming conventions shall be used for all variables declared as CORBA objects.

Example:

```
CF::File * file_ptr           //CORBA Object Pointer type
CF::File var file_var         //CORBA Object var type
CF::File file_obj             //CORBA Object base type
CF::File_in file_in           //CORBA Object in type
CF::File_out file_out         //CORBA Object out type
CF::File_inout file_inout     //CORBA Object inout type
```

Xerces Parser Naming Conventions

In order to differentiate Xerces Parser objects from C++ objects, the following naming conventions shall be used for all variables declared for XML parser classes:

Example:

```
DOMElement* localfileElement // DOM Element Object Pointer type
DOMNode* localfileNode        // DOM Node Object Pointer type
DOMNodeList* localfileNodeList // DOM Node List Object Pointer type
const XMLCh* localfile_xmlch  // XMLString Object Pointer type
```

The first part of the declared variable above is the tag name and the second part is the object pointer type for that particular tag.

Example:

```
Tag: <value> 3 </value>
DOM Element Object Pointer type:    DOMELEMENT* valueElement
DOM Node Object Pointer type:       DOMNode*    valueNode

Tag: <componentfile>
XMLString Object Pointer type:      const XMLCh* componentfile_xmlch
DOM Node List Object Pointer type:  DOMNodeList* componentfileNodeList
```

Variable Usage

Constants

Use of Constants

Avoid the use of numeric values in code. Use symbolic values instead.

Examples:

```
const MAX_VALUE = 5;
for( int i = 0; i < MAX_VALUE; ++i ) // Good Practice

for( int i = 0; i < 5; ++i )         // Poor Practice
```

Constant definitions

Constants shall not be defined in the preprocessor with the `#define`. Constants will be defined using the `const` or `enum` programming definitions.

Examples:

```
const static unsigned int MAX_FILES = 20; // Good Practice

#define MAX_FILES 20;                     // Poor Practice
```

Variables

Variable Initialization

Every numeric variable that is declared shall be initialized with a value before it is used.

Variable Declaration

Variables shall be declared at the beginning of the functions or classes.

Re-declaration of Variables

A local variable shall not be re-declared within a function.

Array Constants

An Array shall not be dimensioned to a hard-coded constant.

Example:

```
int intArray[totalMonths_const]; // Good Practice

int intArray[13];                 // Poor Practice
```

Preprocessor Directives

Multiple Instances of Include Files

Header files shall include code to prevent multiple instances of include files. For example, all data in a header file may be wrapped with the following preprocessor statements:

Example:

```
#ifndef __MY_INCLUDE_H__
#define __MY_INCLUDE_H__
.
.
.
#endif
```

Location of Include Files

Only those #include files necessary for the compilation of a header file shall be included in the header (*.h) files. Include files shall be placed in the *.cpp files whenever possible.

#include comments

A short comment shall be placed with the include file stating the reason the file is being included.

#include directives

#include directives <> shall be used for files containing standard system header information. Double quotes, "", shall be used for include files written by the developer or non-standard system header information.

General Formatting

Arithmetic Parenthesis

Use parenthesis to clarify the order of evaluation for operators in arithmetic expressions.

Examples:

```
i = (( a * b ) + 5 );           //Good Practice
j = ((( a + b ) * 10 ) / 2 );   //Good Practice

x = a + b * 10 / 2;             //Poor Practice
```

Floating Point Declarations

When specifying a constant floating point number, place a number on either side of the decimal point.

Examples:

```
float a = 0.3;                  // Good Practice

float a = .3;                   // Poor Practice
```

Statements

All condition blocks shall be bracketed regardless of the number of body statements. Comment all null statements.

Example:

```
if(condition)
{
    A single statement;
}
while(Workingcondition)
{
```

```

    ;//null
}

```

Switch Statements

Default Branch

A switch statement shall always contain a default branch.

Fall Through

A case statement that falls through to the next case statement shall have a comment stating so.

Example:

```

switch (c)
{
    case 1:
    {
        // fall through
    }
    case 2:
    {
        break;
    }
    default:
    {
        break;
    }
} // end switch

```

Defect Avoidance

Required Methods for a Class

Classes should implement the following methods. If you don't have to define and implement any of the "required" methods, they should still be represented in your class definition as comments. Standardized 'editor' templates for class declaration & definition shall be defined for each project.

- Default Constructor - If your class needs a constructor, make sure to provide one. You need one if during the operation of the class it creates something or does something that needs to be undone when the object dies. This includes creating memory, opening file descriptors, opening transactions etc. If the default constructor is sufficient, add a comment indicating that the compiler-generated version will be used.
- Virtual Destructor - If your class is to be derived from other classes, then make the destructor virtual.
-

Direct Access to Data Members

Direct access to an object's data members shall be avoided. Data members shall not be declared as public. Public accessor methods shall be provided.

Return Values

Public functions shall never return a pointer or references to a local variable. This violates the concept of data encapsulation. Always check the return value of functions that return a pointer.

Function Prototypes

A function prototype and the function definition shall use the same names for their formal arguments.

Example:

```
char * profile (CORBA::Environment &ACE_TRY_ENV);           // prototype
char * Application::profile (CORBA::Environment &ACE_TRY_ENV) // definition
{
}
```

Specification of Return Type

Always specify the return type of a function or member function (exceptions to this rule are the constructors and destructors of a class).

Example:

```
void function1(int i); // Good Practice

function1 (int i);    // Poor Practice
```

Globals

Global data shall be avoided whenever possible.

Macros

Avoid using macros when the same functionality can be implemented using a function. Macros may have unwanted side effects.

Parameter Checking

Each function that is passed arguments shall check the integrity of the arguments before using them.

Code Review Checklist

The following contains a checklist used for code inspections on the OrcaCF. When performing code review this section is printed and used to review each source code file.

General Information and Requirements

Date: _____
Reviewer(s): _____
Developer(s): _____
File Name(s): _____
Test File Name(s): _____ To check Requirements Only:
File Type: {circle 1 or more} Auto / Auto_mod / Reuse_mod / New / Header / Test

Have the requirements been met for the current file under review? ☐yes ☐no ☐n/a ☐c (corrected)
If not give a brief explanation as to why?

SCA Requirements Documented in Header File: _____

SCA Requirements tested: _____

SCA Requirements NOT Documented in Header File: _____

SCA Requirements NOT tested: _____

Questions for the developers? _____

Documentation

3.1	Standard Header	Does the standard header meet the style guide?	<input type="checkbox"/> yes <input type="checkbox"/> no <input type="checkbox"/> n/a <input type="checkbox"/> c
3.2	Requirements	Are the requirements delineated with correct comments?	<input type="checkbox"/> yes <input type="checkbox"/> no <input type="checkbox"/> n/a <input type="checkbox"/> c
3.3	Standard Delineator	File Do all files have a comment on the first line indicating type of source code file?	<input type="checkbox"/> yes <input type="checkbox"/> no <input type="checkbox"/> n/a <input type="checkbox"/> c
3.4	Modifications Delineator	Are modifications delineated per the style guide?	<input type="checkbox"/> yes <input type="checkbox"/> no <input type="checkbox"/> n/a <input type="checkbox"/> c
3.5	*.h Files	Does the documentation of the header files describe the functioning of the class and functions?	<input type="checkbox"/> yes <input type="checkbox"/> no <input type="checkbox"/> n/a <input type="checkbox"/> c
3.5.1	Functions	Are all functions wrapped with asterisks '*' that are 78 characters long	<input type="checkbox"/> yes <input type="checkbox"/> no <input type="checkbox"/> n/a <input type="checkbox"/> c
3.5.1.1	SCA classes and Functions	Do the classes and functions taken from the SCA appear in front of the section they pertain to (including sect. and par. #'s)	<input type="checkbox"/> yes <input type="checkbox"/> no <input type="checkbox"/> n/a <input type="checkbox"/> c
3.5.1.2	Non-SCA Functions	Are the functions not defined in the SCA documented like those taken from the SCA?	<input type="checkbox"/> yes <input type="checkbox"/> no <input type="checkbox"/> n/a <input type="checkbox"/> c
3.6	*.cpp Files	Does the documentation for the *.cpp file describe how the function works?	<input type="checkbox"/> yes <input type="checkbox"/> no <input type="checkbox"/> n/a <input type="checkbox"/> c
3.6.1	Function Delineators	Are the start and ending of functions clearly documented per the style guide?	<input type="checkbox"/> yes <input type="checkbox"/> no <input type="checkbox"/> n/a <input type="checkbox"/> c
3.7	Branching statements	Are the nested blocks of code and blocks exceeding 10 lines clearly defined? (// end if, // end else, // end do)	<input type="checkbox"/> yes <input type="checkbox"/> no <input type="checkbox"/> n/a <input type="checkbox"/> c
	Overall Documentation	Does the overall comments accurately describe what the program should or should not do?	<input type="checkbox"/> yes <input type="checkbox"/> no <input type="checkbox"/> n/a <input type="checkbox"/> c

Notes:

White Space

4.1	Line Length	Are the line lengths within the 78 character limit?	<input type="checkbox"/> yes <input type="checkbox"/> no <input type="checkbox"/> n/a <input type="checkbox"/> c
4.2	Tabs	Are spaces used in place of tab characters?	<input type="checkbox"/> yes <input type="checkbox"/> no <input type="checkbox"/> n/a <input type="checkbox"/> c
4.3	Indentation Length	Are the indentation lengths 3 characters? (textpad paragraph markers)	<input type="checkbox"/> yes <input type="checkbox"/> no <input type="checkbox"/> n/a <input type="checkbox"/> c
4.4	Indent Style	Is the code indented and lined up with the bracket?	<input type="checkbox"/> yes <input type="checkbox"/> no <input type="checkbox"/> n/a <input type="checkbox"/> c

4.5	Statements	Is there only 1 statement per line?	<input type="checkbox"/> yes <input type="checkbox"/> no <input type="checkbox"/> n/a <input type="checkbox"/> c
4.6	Operators	Is there at least 1 space before and after each operator?	<input type="checkbox"/> yes <input type="checkbox"/> no <input type="checkbox"/> n/a <input type="checkbox"/> c
4.7	Variables	"Are variables separated by a ', ' (comma space)?"	<input type="checkbox"/> yes <input type="checkbox"/> no <input type="checkbox"/> n/a <input type="checkbox"/> c
4.8	Argument List	Are function call parameter lists separated by a ', ' (comma space)?	<input type="checkbox"/> yes <input type="checkbox"/> no <input type="checkbox"/> n/a <input type="checkbox"/> c
4.9	Long Statements	In statements where there are 2 or more lines, are 3 space indentations used to show a continuation of the previous line?	<input type="checkbox"/> yes <input type="checkbox"/> no <input type="checkbox"/> n/a <input type="checkbox"/> c
4.10	Declare Parameter List	Are parameters listed below function?	<input type="checkbox"/> yes <input type="checkbox"/> no <input type="checkbox"/> n/a <input type="checkbox"/> c
	Overall White Space	"Overall, is there sufficient white space for readability?"	<input type="checkbox"/> yes <input type="checkbox"/> no <input type="checkbox"/> n/a <input type="checkbox"/> c

Notes:

Naming Conventions

5.1	Names	Do the variable names suggest usage or are they commented correctly?	<input type="checkbox"/> yes <input type="checkbox"/> no <input type="checkbox"/> n/a <input type="checkbox"/> c
5.1.1	Class Names	Does the class name describe what it is?	<input type="checkbox"/> yes <input type="checkbox"/> no <input type="checkbox"/> n/a <input type="checkbox"/> c
5.1.2	Variables	Do all the variables begin with lowercase letters?	<input type="checkbox"/> yes <input type="checkbox"/> no <input type="checkbox"/> n/a <input type="checkbox"/> c
5.1.2.1	Multiple Word Names	Are multi-word names indicated with uppercase letters?	<input type="checkbox"/> yes <input type="checkbox"/> no <input type="checkbox"/> n/a <input type="checkbox"/> c
5.1.2.2	Underscores	Do all variables start with a letter? As opposed to a '_'.	<input type="checkbox"/> yes <input type="checkbox"/> no <input type="checkbox"/> n/a <input type="checkbox"/> c
5.1.2.3	Prefix Notation	Are all variables pre-fixed with the proper identifiers?	<input type="checkbox"/> yes <input type="checkbox"/> no <input type="checkbox"/> n/a <input type="checkbox"/> c
5.1.3	Methods/Functions	Do the names of the methods/functions describe the action clearly?	<input type="checkbox"/> yes <input type="checkbox"/> no <input type="checkbox"/> n/a <input type="checkbox"/> c
5.1.4	Enum names	Are enumerations uppercase with '_' separators?	<input type="checkbox"/> yes <input type="checkbox"/> no <input type="checkbox"/> n/a <input type="checkbox"/> c
5.2.5	CORBA Suffixes	Are the CORBA suffixes indicated?	<input type="checkbox"/> yes <input type="checkbox"/> no <input type="checkbox"/> n/a <input type="checkbox"/> c

Notes:

Variable Usage

6.1.1	Use of Constants	"Are numbers avoided in looping structures?(ie...For, Do, While)"	<input type="checkbox"/> yes <input type="checkbox"/> no <input type="checkbox"/> n/a <input type="checkbox"/> c
6.1.2	Constant definitions	Are #define constants avoided?	<input type="checkbox"/> yes <input type="checkbox"/> no <input type="checkbox"/> n/a <input type="checkbox"/> c
6.2.1	Variable Initialization	Are all numeric variables initialized?	<input type="checkbox"/> yes <input type="checkbox"/> no <input type="checkbox"/> n/a <input type="checkbox"/> c
6.2.2	Variable Declaration	Are all variables declared at the beginning of functions or classes?	<input type="checkbox"/> yes <input type="checkbox"/> no <input type="checkbox"/> n/a <input type="checkbox"/> c
6.2.3	Re-declaration of Variables	Are variables unique and only declared once?	<input type="checkbox"/> yes <input type="checkbox"/> no <input type="checkbox"/> n/a <input type="checkbox"/> c
6.3	Array Constants	Are arrays dimensioned to a variable?	<input type="checkbox"/> yes <input type="checkbox"/> no <input type="checkbox"/> n/a <input type="checkbox"/> c

Notes:

Preprocessor Directives

7.1	Multiple Instances of include files	Is the header file wrapped with pre-processor statements to prevent multiple instances of include files?	<input type="checkbox"/> yes <input type="checkbox"/> no <input type="checkbox"/> n/a <input type="checkbox"/> c
7.2	Location	Are the include files located in the *.cpp file as specified by the style guide?	<input type="checkbox"/> yes <input type="checkbox"/> no <input type="checkbox"/> n/a <input type="checkbox"/> c
7.3	# include Comments	Is the include file commented properly?	<input type="checkbox"/> yes <input type="checkbox"/> no <input type="checkbox"/> n/a <input type="checkbox"/> c
7.4	#include Directives	Are the proper directives indicated?	<input type="checkbox"/> yes <input type="checkbox"/> no <input type="checkbox"/> n/a <input type="checkbox"/> c

Notes:

General Formatting

8.1	Arithmetic Parenthesis	Are parenthesis used to clarify order of evaluation?	<input type="checkbox"/> yes <input type="checkbox"/> no <input type="checkbox"/> n/a <input type="checkbox"/> c
8.2	Floating Point Declarations	Are numbers placed on either side of decimal point?	<input type="checkbox"/> yes <input type="checkbox"/> no <input type="checkbox"/> n/a <input type="checkbox"/> c
8.3	Statement	Are all the condition blocks bracketed?	<input type="checkbox"/> yes <input type="checkbox"/> no <input type="checkbox"/> n/a <input type="checkbox"/> c
8.4.1	Default Branch	Do all switch statements contain default branches?	<input type="checkbox"/> yes <input type="checkbox"/> no <input type="checkbox"/> n/a <input type="checkbox"/> c
8.4.2	Fall Through	Are all fall throughs commented?	<input type="checkbox"/> yes <input type="checkbox"/> no <input type="checkbox"/> n/a <input type="checkbox"/> c

Notes:

Defect Avoidance

- | | | | |
|-----|---------------------|--|--|
| 9.1 | Required Methods | Are the required constructors and destructors included? | <input type="checkbox"/> yes <input type="checkbox"/> no <input type="checkbox"/> n/a <input type="checkbox"/> c |
| 9.2 | Data members | "Is access to public, virtual or protected data members avoided? (are access functions created and/or private sections included?)" | <input type="checkbox"/> yes <input type="checkbox"/> no <input type="checkbox"/> n/a <input type="checkbox"/> c |
| 9.4 | Function Prototypes | Are the same names used for function prototype and definition? | <input type="checkbox"/> yes <input type="checkbox"/> no <input type="checkbox"/> n/a <input type="checkbox"/> c |

Notes:
